Andrea Marseglia, Tomi Rantanen

Prosessin suunnittelu lokiseurannan QlikView-näytön seurantaan

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

28.11.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Andrea Marseglia, Tomi Rantanen Prosessin suunnittelu lokiseurannan QlikView-näytön seurantaan 55 sivua 28.11.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Simo Silander Senior Service Manager Jani Niskanen

Insinöörityön tavoitteena on suunnitella ja jalkauttaa prosessi IBM MQ -lokiseurannan QlikView-näytön seurantaan. Tuloksena saadaan jalkautettua kahden tiimin päivittäiseen toimintaan ennakoivaa virheenhallintaa parantava prosessi, jonka myötä sanomapohjaisen väliohjelmiston muodostamien lokikirjauksia monitoroivan QlikView-näkymän lukeminen ja siitä tehtävät johtopäätökset mukautetaan osaksi teknisen hallinnan tiimien jokapäiväistä toimintaa.

Insinöörityö koostuu QlikView-näytön toiminnan suunnittelusta, lokiseurantaprosessin käyttötarkoituksen ideoinnista, prosessin toteutuksen suunnittelusta, seurantaprosessin toteutuksesta, prosessin jalkautuksesta loppukäyttäjille ja valmiin prosessin käytettävyyden arvioinnista loppukäyttäjien näkökulmasta.

Työssä esitellään järjestelmäintegraatiot, jatkuvien palveluiden teknisen hallinnan tiimin toiminta, sanomapohjaisen valvonnan perusperiaatteet, erilaisten valvontatyökalujen hyödyntäminen valvonnassa ja muutamia valvontaprosesseja, joita käytetään osana suunniteltua prosessia.

Insinöörityön tavoitteet saavutettiin suurimmilta osin: näkymän ja prosessin suunnittelu saatettiin onnistuneesti loppuun ja prosessia pilotoitiin oikeaa kohderyhmää vastaavalla testiryhmällä, mutta varsinainen jalkautus ei insinöörityön aikataulun puitteissa onnistunut. Jalkautus tullaan tekemään annettujen ohjeiden mukaisesti lähitulevaisuudessa.

Avainsanat	Järjestelmäintegraatio, QlikView, Logstash, IBM MQ, Prosessi, Kehitys, Suunnittelu, Point-to-Point, Palveluväylä, EAI, Valvonta

Author(s) Title Number of Pages Date	Andrea Marseglia, Tomi Rantanen Process Development for Monitoring QlikView-Implemented Eventlog Manager 55 pages 28 November 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Jani Niskanen, Senior Service Manager Simo Silander, Senior Lecturer

The aim of this thesis was to design, develop and implement a process to monitor a QlikView-implemented eventlog manager. As a result, a pre-emptive process for enhanced incident management was implemented for daily use in two separate teams in the technical management of the continuous services.

The process was designed to visualize the filtered and analyzed eventlog generated by the messages passing through a message-oriented middleware. The eventlog was visualized on a QlikView-monitor in a way that it is fast and easy for a member of either of the teams to make unambiguous decisions about the status and state of the integration.

This study consisted of designing the functionality of the QlikView-monitor, designing the use of the eventlog filtering process, producing the monitoring process, the implementation of the process to the two teams and designing on how the series of processes function as a whole. The produced process was then reviewed by the members of each team.

The study provides an introduction to the system integration, the basis on how a technical management team operates in the continuous services, the basic principles of the message-oriented monitoring and the utilization of monitoring tools in the monitoring process. Also the monitoring processes used in this particular process are discussed.

Keywords	Systems integration, QlikView, Logstash, IBM MQ, Process, Development, Design, Point-to-Point, ESB, EAI, Monitoring

Sisällys

Lyhenteet

QV	QlikView, Qlik-ohjelmistotalon kehittämä liiketoimintatiedon hallintajärjestelmä.
XML	Extensible Markup Language, rakenteellisen merkityksen kuvauskieli.
IBM	International Business Machines, teknologiayritys, joka tunnetaan muun muassa suurtietokoneiden ja raskaiden palvelimien valmistajana.
ITIL	Information Technology Infrastructure Library, prosessikehys, eli käytäntöjen kokoelma IT-palveluiden johtamiseen ja hallintaan.
EAI	Enterprise application integration, sanomaintegraation palveluväylä.
MQ	IBM MQ, Message Queue, IBM:n kehittämä ratkaisu sanomanvälitykseen.

1 Johdanto

Teknologiayrityksen päivittäiseen toimintaan kuuluu toisistaan riippumattomina suoritettavia toimenpiteitä. Kun nämä toimenpiteet suunnitellusti toteuttavat halutun lopputuloksen, niistä muodostuu prosesseja.

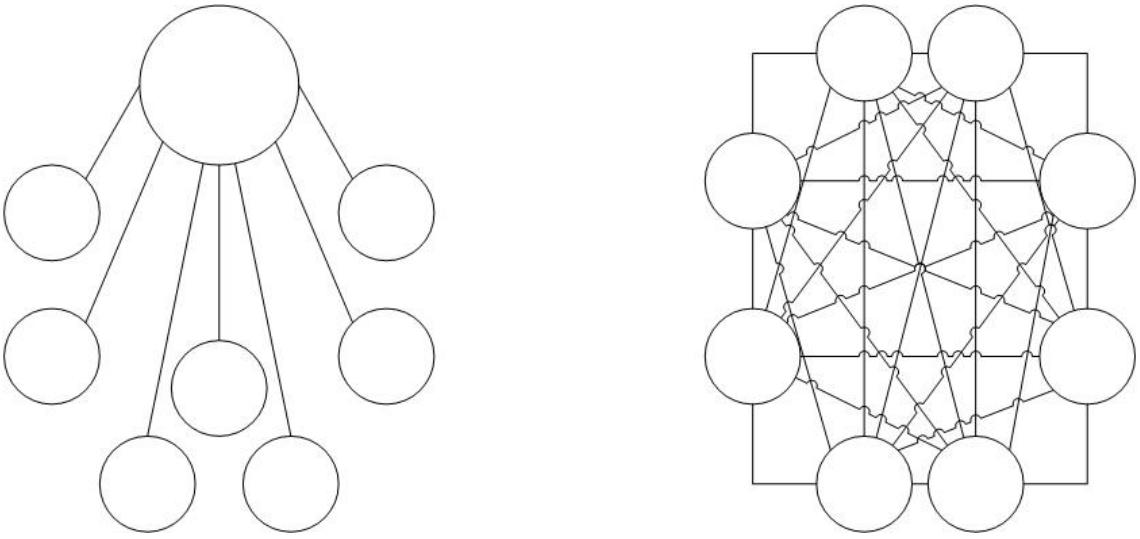
Prosessit voivat olla alati toiminnassa tai käynnistyä tarpeen vaatiessa tiettyjen olosuhteiden tai kriteerien täytyttyä. Toteutusmallista riippumatta prosesseja hyödynnetään niin isoissa projekteissa, pienissä sprinteissä, yksittäisissä työnannoissa kuin tiimien päivittäisessä toiminnassa. Mikä tahansa ympäristö tai toiminto on prosessin alainen eli prosessoitu, mikäli hyväksi havaittujen tai käskettyjen suoritesarjojen toteuttaminen on välttämätöntä parhaan lopputuloksen saavuttamiseksi.

Teknologiayritys koostuu useista pienistä osasista, tiimeistä, joilla on jokaisella oma tarkoituksensa. Tiimit eivät välttämättä ole jokainen konkreettisesti oma erillinen kokonaisuutensa vaan, ne jakautuvat limittäin yrityksen työntekijöiden kesken rooleiksi. Yksi työntekijä voi siis olla monen eri tiimin jäsen ja omata niin esimies- kuin kehittäjärooleja. Teknologiayrityksessä tiimit voivat toimia eri periaattein ja pyrkiä eri lopputulokseen. Näin ollen työntekijältä vaaditaan usean eri työtavan, -prosessin ja -kulttuurin omaksumista. Omaksuttaessa uutta roolia on tärkeää, että myös prosessikuvaukset ovat helposti omaksuttavissa, eli ne eivät eroa liikaa yrityksen sisällä. Lisäksi niiden tulee olla helposti saatavilla, dokumentoituina.

Tiimien työnkuvat ja niille jalkautetut prosessit eroavat toisistaan usein tiimin tavoitteiden takia (kuva 1). Sovelluksien ja ohjelmistojen kehitystiimit pyrkivät suunnittelemaan ja kehittämään uutta ja lisäksi prosessoimaan toteuttamiaan tuotteita suunnitellusti ja saadun palautteen avulla. Myyntitiimit pyrkivät tuomaan ja levittämään julkisuutta sovellukselle sekä kapseloimaan kehitetyn sovelluksen asiakkaalle helposti ymmärrettävään muotoon. Jatkuvien palveluiden ongelmanratkaisu- ja valvontatiimit puolestaan pyrkivät ylläpitämään sovellukselle tuotetut yhteydet ja varmistamaan asiakastyytyväisyyden valvomalla muodostettujen yhteyksien toimintaa sekä toimimalla asiakaspalautteen prosessoivana linkkinä kehitystiimille. Kehitystiimien päivittäin käyttämät prosessit koskevat muun muassa toimintaa eri ympäristöissä, versionhallintaa ja yhtenäistä tapaa kirjoittaa lähdekoodia. Myyntitiimeillä prosessit ovat sosiaalista tuntemusta ja sosiaalipsykologian

tietämykseen perustuvia – miten esittää asia erilaiselle ihmistyypille, miten ohjata asiakkaan tarkkavaisuuden kohdetta esitettäessä asiaa. Tuote ei ole myyntitiimin mukaan koskaan puutteellinen, mutta sen kehitys on vain jatkuvaa. [1. s.178]

Jatkuvien palveluiden valvontatyössä on käytössä useita työkaluja ja prosesseja työkalujen käyttöön. Valvontatyön keskeinen osa on sanomapohjaisen liikenteen valvonta, mikä koostuu palvelinyhteyksien valvonnasta, integraatiototeutuksissa kulkevien sanomien valvonnasta ja sanomaliikenteen jatkuvuuden valvonnasta. Valvontatyö on usein keskitettyä, jolloin useita asiakkaita palvelevan organisaation asiakastoteutukset voivat olla eriäviä, yksittäisen toteutuksen riippuessa asiakkaan vaatimuksista ja asiakkaan sidosryhmien käyttämien järjestelmien vaatimuksista. Valvontatyön laadun takaamiseksi ja havaittujen ongelmatilanteiden nopean ratkaisun kannalta tulee ylläpitää palvelutietämyksen hallintajärjestelmä, johon on kirjattu asiakaskohtaiset tiedot rakennetuista yhteyksistä ja jonne kirjataan talteen ongelmatilanteiden ratkaisukuvaukset virhetilanteen seuraavaa esiintymistä varten. Tiedon leviäminen eri käyttäjien kesken on jokaisessa tiimissä ehdottoman tärkeää, sillä muutoin ongelmatilanteen esiintyessä uudelleen käytetään toistuvasti aikaa tilanteen analysointiin ja mahdollisen ratkaisun selvittämiseen sen sijaan, että tilanteen kuvaus ja ratkaisu olisi välittömästi saatavilla. Pelkkä palvelutietämyksen hallintajärjestelmän olemassaolo ei hyödytä tiimin valvontatyötä, vaan myös järjestelmän päivittäinen käyttäminen ja täydentäminen tulee olla prosessoitua.



Kuva 1. ”Chief”-ohjelmointitiimin rakenne (vas.) ja ”Egoless”-ohjelmointitiimin rakenne. ”Chief” on yleisesti käytössä sovelluskehityksessä ja sen ”päämiehenä” toimii tiimin vanhin, joka on vastuussa teknillisistä päätöksistä ja tiimin jokaisella osasella on oma roolinsa. ”Egoless” on etenkin teknisten hallintatiimien omaksuma rakenne, missä jokaisella osalla on toista vastaava tietämys, jolloin saavutetaan yhtäläinen laatu niin ratkaisutyössä kuin asiakasrajapinnassa. [2. s.178-179]

Tässä insinöörityössä keskitymme Digia Finland Oy:n jatkuvien palveluiden tiimin toimintaan. Tavoitteena on suunnitella, luoda ja jalkauttaa tiimin tuki- ja valvontatyössä hyödynnettävä prosessi lokiseurannan QlikView-näkymän seurantaan. Prosessi suunnitellaan kuvaamaan näytölle annetun kohteen proaktiiviset prosessit, joilla käytetyn alustan ja infrastruktuurin virhetilanteet on mahdollista arvioida jo ennalta, ennen kyseisen virhetilanteen realisoitumista. Näytölle kuvattu tilanne on tiimille asiakkuuden viestiliikenteen tilannekuva sekä samalla sanomaintegraation palveluväylän kuormituksen tilannekuva sekä työkalu resursseja vaativien ongelmatilanteiden ja laajojen häiriötilanteiden ennaltaehkäisemiseen. Jo yksittäinen laaja häiriö estää liiketoimintaa kahdella tavalla:

1. Laaja häiriö muodostaa niin sanotun patouman muulle sanomaliikenteelle, jolloin sanomaliikennettä lähettäjän ja vastaanottajan välillä ei ole mahdollisesti lainkaan. Tilanne voi eskaloitua jo aiemmin toteutetuilla integraatioilla useiden eri virhetilanteiden yhtäaikaisen toiminnan seurauksena tai esimerkiksi integraatiototeutuksen päivityksen yhteydessä tai uuden integraation käynnistämisen seurauksena. Laajan häiriön voi aiheuttaa myös esimerkiksi jokin luonnonmullistus, jolloin sähkökatkoksen takia yhteyttä ei ole lainkaan saatavilla.

2. Laajasta häiriöstä toipuminen vie resursseja valvontatiimiltä, jolloin esimerkiksi muu integraatiototeutuksien kehitystyö on selvitystyön takia jäissä. Häiriöstä toipuminen vie aikaa ja aiheuttaa mahdollisesti palvelukatkoksia asiakkaille, jotka eivät ole kannattavalle liiketoiminnalle edullisia.

Optimitilanteessa näytölle tuotettu data toimii nollailmoituksena, eli ilmoituksena seurattavan ympäristön virheettömästä tilasta, jolloin tiimi voidaan resursoida keskittymään jonovalvonnan sijaan muualle.

Sanomaliikenne toteutetaan etenkin suuren kapasiteetin liittymissä käyttäen viestipohjaista jonomanageria. Sanomat lähetetään päätepisteen sijaan odottamaan jatkokäsittelyä jonosovellukseen, josta niitä käydään lukemassa asynkronisesti sovellukseen lähetetyn kutsun avulla. Näin kontrolloidaan viestien järjestystä ja tasapainotetaan kuormitusta. Jonomanagerin kaikesta toiminnasta kirjoitetaan lokitapahtumia. ”Jono käynnistetty”, ”vastaanotettu sanoma”, ”haettu sanoma”, ”jonon maksimisyvyys saavutettu” ja ”jonossa olevan viestin tietokentät ovat korruptoituneet” ovat esimerkkejä lokitapahtumiin kirjatuista tapahtumista. Työssä hyödynnetään näitä viestipohjaisen jonomanagerin tuottamia tapahtumalokeja, joita luetaan erillisen sovelluksen avulla. Viestijonoja on useita

ja niiden läpi kulkee yhteensä jopa miljoonia sanomia päivässä. Näiden sanomien aiheuttamien prosessointitapahtumien kirjoittamat lokitapahtumat luetaan, prosessoidaan ja visualisoidaan erilliselle näkymälle, josta käy helposti ilmi mahdolliset vikatilanteet ja optimitilanteessa pelkkä nollailmoitus. Täten tiimin on mahdollista selvittää asiakkuuden viestiliikenteen tilanne ja nopeasti siirtyä seuraaviin työtehtäviin suorittamatta laajempaa selvitystyötä. [1. s,152-154.]

Lopputuloksena saadaan aikaan kahden eri tiimin, tuki- ja suunnittelutiimin valvontatyöhön jalkautettu, päivittäistä toimintaa ohjaava käytäntö. Käymme työssä läpi prosessin kehitysvaiheet aina jalkauttamisen jälkeiseen arviointiin asti ja vertailemme saavutettuja tuloksia suhteessa ohjelmistotalon omaa työkalua vasten kuin eri tiimien loppukäyttäjien näkökulmista. Prosessin suunnittelussa otetaan huomioon niin työn tilaajan kuin kohdetiimien tarpeet ja tavoitteellinen suunnittelu tehdään yhteistyössä ohjelmistotalon analytiikkaosaston kanssa.

2 Järjestelmäintegraatio

Sinitarra on öljystä, kumista ja polymeereista koostuva monikertakäyttöinen liimatahna. Kaikkeen, mitä voi koskettaa, voi laittaa sinitarraa. Kun A4-kokoisen paperin painaa vasten seinää ja paperin ja seinän väliin jää riittävä määrä sinitarraa, paperi kiinnittyy seinään. Sinitarra toimii väliaineena kiinnittyneen artikkelin ja seinän välillä. Samalla luodaan seinälle kirjoitusrajapinta tärvelemättä itse seinää: seinää katselemalla voidaan nähdä paperille kirjatut tiedot. Näin luodaan reaalimaailman integraatiototeutus. Yksinkertaisimmillaan integraatio toimii kuin sinitarra. Digitaalisessa maailmassa esimerkiksi Excel-taulukkotiedoston ja minkä tahansa rajapinnan välille voidaan alkeellisesti ”laittaa sinitarraa”, jolloin Excel-tiedostoon kirjatut muutokset siirtyvät rajapintaintegraation lävitse nähtäville kohteeseen. Järjestelmäintegraatiototeutus on kuitenkin yksittäistä liityntäkohtaa monimutkaisempi ja palvelee useampaa kuin yhtä käyttötarkoitusta.

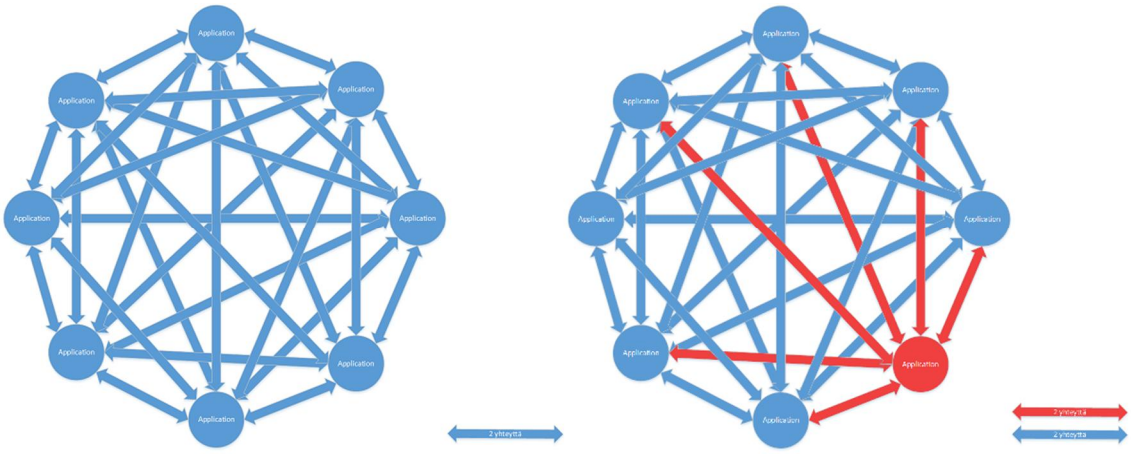
Järjestelmäintegraatiolla tarkoitetaan tapaa yhdistää kokonaisia yksittäisiä järjestelmiä ja sovelluksia isommaksi kokonaisuudeksi. Järjestelmäintegraatiossa eri järjestelmien tuottama data muunnetaan keskenään yhteensopivaksi. Kyseinen toimintatapa yhdistää organisaation erillisiä järjestelmiä on tullut yhä tärkeämmäksi alati kehittyvässä digitalisoituvassa maailmassa, jossa vaihtoehtoja eri rakenneosiksi on useita. Osat eivät vält-

tämättä ole suoraan yhteensopivia, vaan vaativat väliohjelmiston toimiakseen. Järjestelmäintegraatiolla datan siirtyminen järjestelmien välillä saadaan hoidettua nopeasti ja valvotusti.

Järjestelmäintegraatioita voidaan rakentaa eri tavoilla riippuen kyseessä olevan projektin tarpeista ja vaatimuksista. Pienemmissä kokonaisuuksissa järjestelmäintegraatio voidaan toteuttaa Point-to-Point-integroinnilla. Isommissa kokonaisuuksissa taasen integraatioilta vaaditaan skaalautuvuutta ja pitkässä juoksussa niiden toteuttaminen horisontaalisen integraatioarkkitehtuurin eli palveluväylän avulla on huomattavasti kustannustehokkaampaa. [3.]

2.1 Point-to-Point-integraatio

Point-to-point-integraatiolla tarkoitetaan tapaa, jolla integroitavan kokonaisuuden jokainen järjestelmä yhdistetään toisiinsa erillisillä yhteyksillä (kuva 2). Tämän menetelmän kustannustehokkuus pienemmissä kokonaisuuksissa perustuu siihen, että integraatioiden pohjalla oleva infrastruktuuri on hyvin kevyt ja vähentää näin ollen ylläpitokustannuksia.



Kuva 2. Point-to-point-arkkitehtuuri ja muutosten vaikutus integraatioliittymiin.

Point-to-Point-integraatio on arkkitehtuuriratkaisuna toimiva, kun integroitavia järjestelmiä on vain muutamia kappaleita. Arkkitehtuuri kohtaa kuitenkin ongelmia, kun kokonai-

suuteen lisätään järjestelmiä ja yhteyksien määrä kasvaa eksponentiaalisesti. Esimerkiksi integroitava kokonaisuus sisältää n kappaletta järjestelmiä; tällöin yhteyksiä y_n integraatiokokonaisuudessa on

$$\sum_{y=1}^{n} \quad y_n = nn_{n-1}$$

kappaletta. Kun integraatioon lisätään uusi järjestelmä, yhteyksien lukumäärän kasvu Δy_n seuraa summakaavaa

$$\sum_{y=1}^{n} \quad \Delta y_n = nn_{n-1} - n_{n-1}n_{n-2} \ .$$

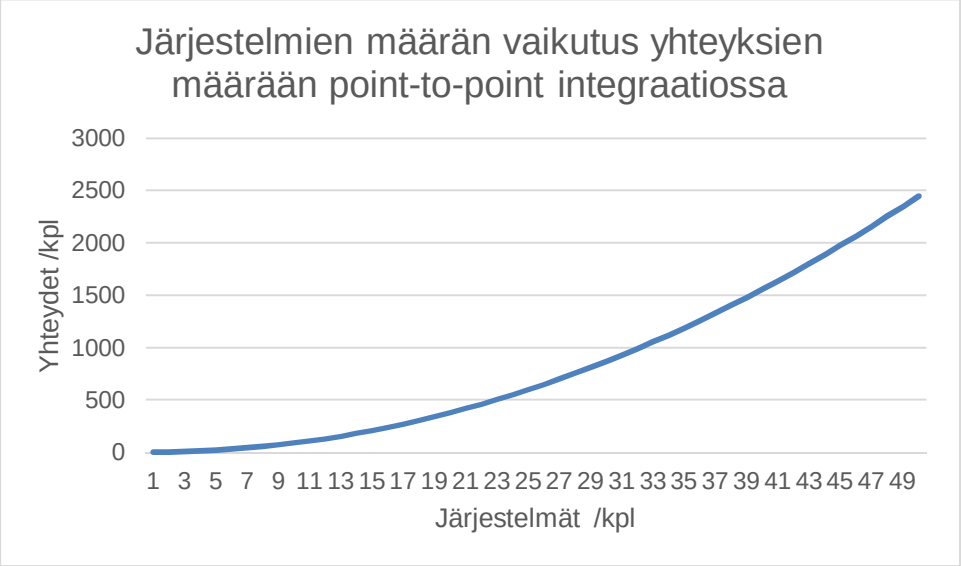
Näin ollen järjestelmien määrän ja yhteyksien määrän erotus muodostaa aritmeettisen jonon, joka määrittää yhteyksien määrän kasvunopeuden kaavalla

$$\sum_{v=i}^{n} \quad v_n = n - (nn_{n-1} - n_{n-1}n_{n-2}) \qquad v_n = n - 2 \qquad .$$

Seuraavassa kuviossa (kuva 3) on havainnollistettu yhteyksien kasvua integroitavien sovellusten lisääntyessä. Kuviosta selviää, että yhteyksien kasvunopeus noudattaa geometristä sarjaa, joka lähestyy lukua 1. Tämä tarkoittaa, että point-to-point-integraation järjestelmien määrän kasvaessa rajatta järjestelmien välisten yhteyksien määrän kasvunopeus ei kasva eksponentiaalisesti vaan voisi automaatiototeutuksella pysyä hallittavissa. Tästä kehittyneemmälle integraatioratkaisulle järjestelmien väliset yhteydet tarkoittavat sanoman lähettäjä-vastaanottaja-parien määrää, joten parien kasvunopeuden lähestyessä lineaarista kasvua niiden monitorointi on myös hallittavissa.

Point-to-point-integraation tarkoitus on kuitenkin olla vain yksittäisratkaisu, eikä kokonainen järjestelmäintegraatio. Integraatiorakenteen kasvaessa liittymien määrä point-to-point-integraatiossa kasvaa käsin tehtäville muutoksille hallitsemattoman suureksi jo kymmenillä järjestelmillä. Lisäksi Point-to-Point-integraatiossa muutoksen hallinta ja

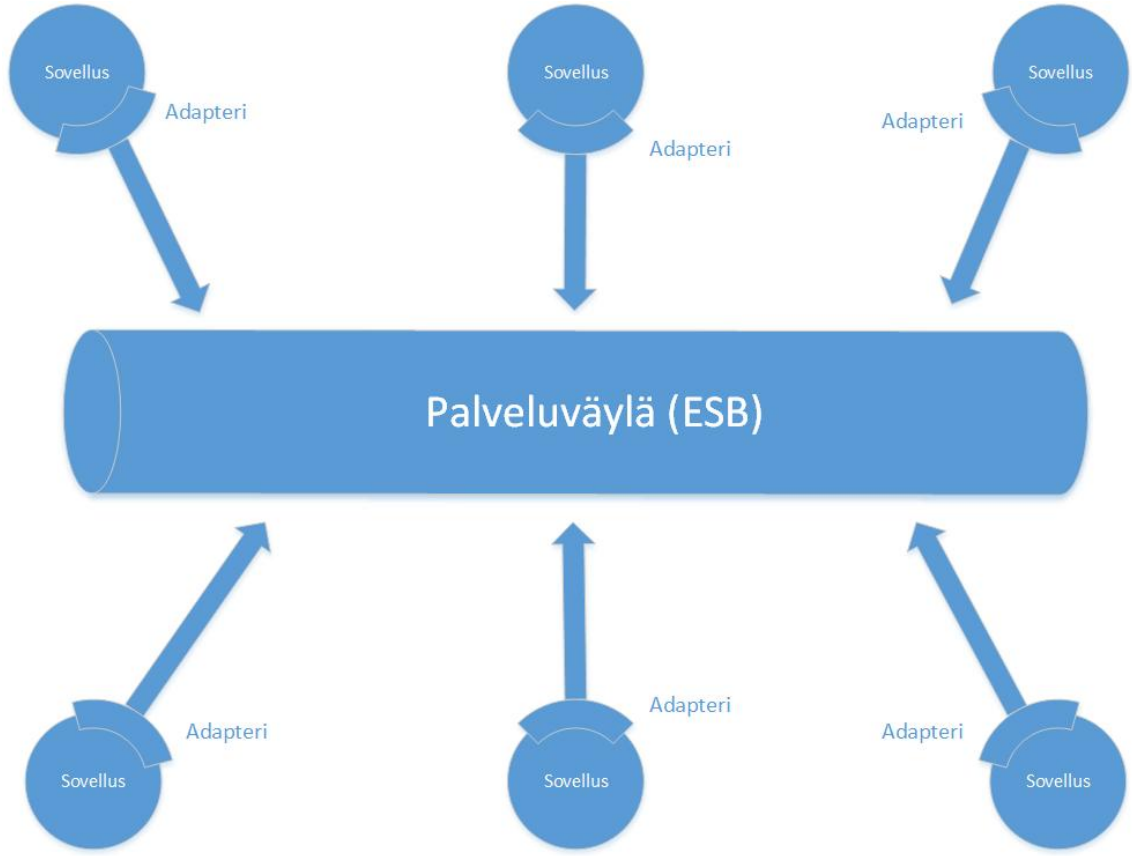
konfiguraation ylläpitäminen ovat huomattavan kalliita ja työläitä muuttaa. Yhteen järjestelmään tehtävä muutos vaikuttaa pahimmassa tapauksessa useaan integraatioon (kuvan 2. oikea puoli) eikä kaikkia vaikutuksia nähdä saman tien. Tässä tapauksessa muutokset on tehtävä erikseen jokaiseen siirtoon, jonka jälkeen siirto on testattava ja tarkistettava, ettei muutos vaikuta muihin olemassa oleviin integraatioihin. [3.]



Kuva 3. Järjestelmien lukumäärän vaikutus yhteyksien määrään Point-to-Point-integraatiossa

2.2 Horisontaalinen integraatio

Toinen tapa toteuttaa järjestelmäintegraatioita on niin sanottu horisontaalinen integraatio (Horizontal Integration). Horisontaalisen integraation perusajatuksena on integraatioiden toteutus palveluväylän (ESB Enterprise Service Bus) avulla. Tarkoituksena on, että jokaiselle integroitavalla järjestelmälle tehdään vain yksi yhteys integraatioväylään (kuva 4). Järjestelmien välisiä suoria integraatioita ei toteuteta, vaan kaikki liikenne kulkee integraatioväylän kautta. Tämä vähentää yhteyksien määrää integraatiokokonaisuudessa ja näin ollen vähentää ylläpidollisia kohtia. Horisontaalinen integraatio toteutetaan usein jonkin käyttötarkoitukseen soveltuvan sovelluksen avulla



Kuva 4. Enterprise service Bus -arkkitehtuuri

Horisontaalisen integraation ylläpidettävyys on kustannustehokasta yllämainitun Point-to-Point-integraatiototeutukseen verrattuna. Horisontaaliseen integraatiototeutukseen tehtävä muutos, joka vaikuttaa vain integraatiokokonaisuuden yhteen sovellukseen, vaikuttaa vain kyseisen sovelluksen ja integraatioväylän väliseen yhteyteen. Näin muutosten tekeminen ei aiheuta suurta määrää selvitystyötä, mikäli muutos tekisi yhteyden käyttökelvottomaksi. Point-to-Point-integraatiototeutuksessa muuttujia on enemmän, jolloin olemassa on useampia mahdollisia yhden muutoksen aiheuttamia virhetilanteita. Yksi muutos voi aiheuttaa aineistollisen virheen myös vasta muutaman onnistuneen integraatiosiirron jälkeen, mikäli muutos vaikuttaa lähettävän sanoman merkistöön. Integraatio voi toimia vuosia kohtaamatta virheen aiheuttamaa merkkiä. [3.]

3 Prosessin kehittäminen

Jatkuvien palveluiden tiimiin on kehitetty useita erilaisia toimintatapoja eli prosesseja, ja ne eroavat usein asiakaskohtaisesti. Jatkuvat palvelut ovat toteutuksen jälkeisiä ylläpito-, tuki- ja valvontapalveluita, joita tarjotaan asiakkaalle palvelusopimuksen mukaisesti.

Prosessien kehittämisessä on kyse hallitusta toimintamallin muuttamisesta. Prosessi koostuu useista toisistaan riippumattomista toiminnoista, joiden suunniteltu toiminta toteuttaa halutun lopputuloksen. Prosessi on käynnistyessään otettu käyttöön suunnitellulla tavalla, mutta sen jälkeen prosessit voivat olla jatkuvan muutoksen alaisina; prosessin suunnitteluvaiheessa on liki mahdotonta kartoittaa jokainen mahdollinen prosessin kohtaama virhetilanne. Pitkäaikaisten asiakkaiden jatkuviin palveluihin jalkautetut prosessit voivat olla vuosienkin jälkeen yhä samanlaisia ja samassa tilassa kuin ensikertaa käynnistyessään.

Usein kohdattaessa prosessin toiminnan kannalta kriittinen virhe on, että prosessin toiminta pyritään varmistamaan nopealla korjauksella. Esimerkkejä tällaisista kriittisistä virheistä on muun muassa se, ettei sidosryhmää informoida riittävän ajoissa tai ollenkaan ja sopimuksen mukaiset vastausajat ylittyvät, koska komentoketjun mukainen varmistus kulkee aikavyöhykkeiden ylitse. Prosessin kehityksessä muutokset keskittyvät organisaation toimintatapoihin sekä tietojärjestelmien kehittämiseen. Siksi nykyisten toimintatapojen ja tietojärjestelmien tilan kartoittaminen on tärkeää, jotta prosessin nykytila saadaan selville ja mahdolliset virhealttiit osat voidaan tuoda esiin. Prosessin kehittämisen tavoitteena onkin kehittää toimintatapoja ja tunnistettujen ongelmakohtien selvitystä.

3.1 Prosessi käsitteenä

Prosessi on suunniteltu, toistettava ja luotettava suoritettavien toimenpiteiden sarja, jonka tuloksena saavutetaan ennalta määritelty lopputulos. Jokainen määrätty tai määräämätön vuorovaikutus reaalisen tilanteen kanssa voidaan nähdä prosessina: kahvinkeittimen vesisäiliön täyttäminen, suodatinpussin asettaminen, kahvipurujen annostelu ja kahvinkeittimen virtanappulan painaminen ovat esimerkkejä kahvinkeittoprosessista. Prosessin kohtaamia kriittisiä virheitä ovat esimerkiksi, ettei kahvinkeitin ole kytkettynä verkkovirtaan tai tippalukon lukkoasentoon unohtaminen. Kahvipannun keittovaiheessa

sisältämä käsitiskiaine on prosessin kohtaama poikkeus, joka ilman monitoroitua herätettä huomataan vasta kahvia nautittaessa.

Prosessin luomisen tarkoituksena on luoda annetuilla muuttujilla haluttu lopputila: pannullinen kahvia.

Prosessin kehittämisen tarkoituksena on saada halutusta lopputuloksesta paras mahdollinen: pannullinen hyvää kahvia.

Kahvin keittäminen on hyvin käytännönläheinen esimerkki prosessista. Mitä syvemmälle teknologiayrityksen maailmaan mennään, sitä suurempia prosessikokonaisuuksia tulee vastaan. Prosesseihin sisältyy useita prosesseja, jotka taas koostuvat uusista prosesseista.

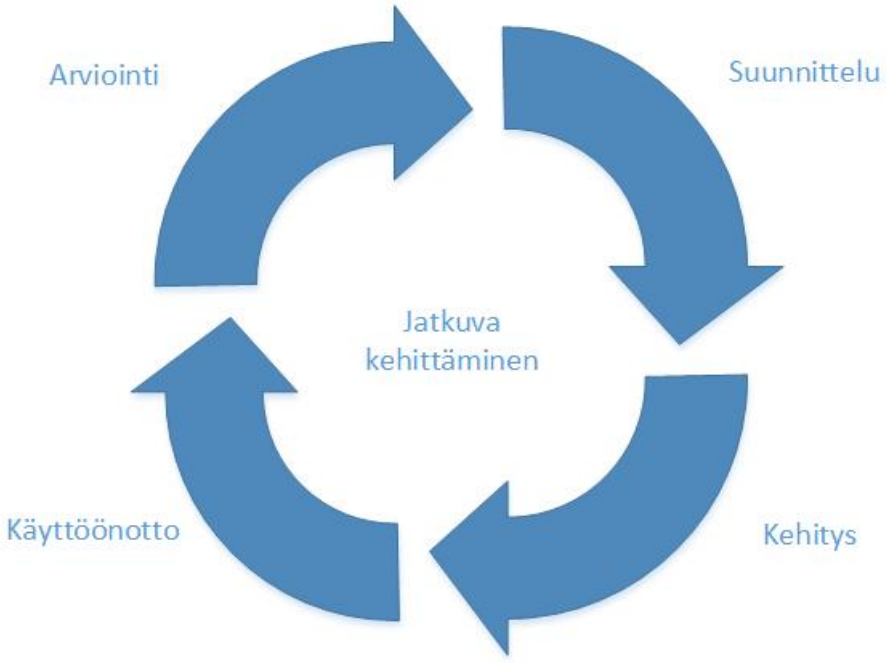
Jatkuvissa palveluissa eri prosesseja sovelletaan muun muassa asiakkaan ympäristöjen valvontaan, asiakkaan viestiliikenteen seuraamiseen, muutoksen vientiin testi- tai tuotantoympäristöön ja uuden toiminnon tuotantoon vientiin. Esimerkiksi tikettijärjestelmään tullut palvelupyyntö ensin huomioidaan ja pyyntö merkitään huomioiduksi, otetaan ajallaan työn alle ja merkitään työn alle otetuksi ja lopulta suoritetaan ja merkitään suoritetuksi. Näin pyynnön tehnyt osapuoli pysyy koko ajan tietoisena pyyntönsä suoritustilasta ja suorituksen aikaleimoista jää merkintä palvelupyyntösopimuksen valvontaan. Kuvattu prosessi mahdollistaa palvelupyyntöjen aikarajojen täyttymisen valvonnan annettuna ajankohtana. Täten voidaan varmistua palvelun toimivuudesta ja saavutetusta laadusta. ICT-maailman ulkopuolelta esimerkkejä luonnonmukaisista prosesseista ovat esimerkiksi nesteen haihtuminen, fotosynteesi ja solujen jakautuminen, teknologisia prosesseja puolestaan muun muassa ydinfissio ja elektrolyysi.

3.2 Prosessin kehittämisen vaiheet

Prosessin kehityksessä käytetään usein jatkuvan kehityksen kehää (kuva 5). Prosessin kehittämisen alkuvaiheessa on selvitettävä kehitettävän prosessin nykytila, mitä mahdolliset ongelmatilanteet ja kehitystarpeet ovat ja mistä ne johtuvat. On selvitettävä syyt sille, miksi kyseistä prosessia halutaan ylipäätänsä kehittää. Prosessin kehittämisen ensimmäisessä vaiheessa on myös syytä asettaa tavoitteet valmiille prosessille. Jo tässä

vaiheessa on hyvä suunnitella valmiin prosessin tavoitteet tarpeeksi tarkasti, jotta prosessin analysointi kehityksen loppuvaiheessa voidaan suorittaa kattavasti vertaillen määrittelyvaiheessa asetettuihin tavoitteisiin. Olennaisena osana prosessin kehitystä on kehityssuunnitelman laatiminen, josta on selvittävä aikataulu, kehitysvaiheet ja niiden prioriteetit sekä tavoitetila.

Ensimmäisen vaiheen jälkeen prosessi kehitetään niin pitkälle, että sen käyttöä voidaan testata kohderyhmää vastaavalla suppeammalla testiryhmällä. Tässä vaiheessa prosessi on vielä puutteellinen ja saattaa sisältää huomattavan määrän ylimääräisiä toimenpiteitä. Täten kaikki testiryhmältä saatava palaute otetaan osaksi prosessin jatkokehitystä. Edellä mainitun pilotoinnin avulla projektitiimi kykenee kehittämään prosessista kustannustehokaan ja tiettyyn tarkoitukseen hyvin soveltuvan toimintatavan.



Kuva 5. Prosessikehityksen kaavio

3.3 Prosessin analysointi

Prosessin analysointi on prosessin peräkkäisten, toisistaan riippumattomien tapahtumien keskinäisen vaikutuksen ja niiden lopputuloksen kannalta olennaisuuden kuvaamista. Prosessin koostuessa useista välivaiheista voi prosessin kehittymisen myötä jostakin välivaiheesta tulla turha tai se voidaan sulauttaa toiseen välivaiheeseen. Prosessianalyysilla saadaan selville, miten haluttuun lopputulokseen päästään ja miten halutut

välivaiheet suoritettiin sekä mikä on prosessin ja sen välivaiheiden merkitys muulle tuotannolle. Analysointi mahdollistaa prosessin heikkojen kohtien paikantamisen ja niiden asettamisen kehityksen alle. Prosessin analysointi voi olla ohjailevaa tai informatiivista.

Ohjaileva, eli preskriptiivinen analysointi on tapahtuman käyttötilanteen tutkimista, eli tutkitaan, millä tavalla toiminto tapahtuu ja miten se on toistettavissa. Esimerkiksi kahvin keittämisessä kiehuva vesi johdetaan kahvipuruille, mikä aiheuttaa liukenemista. Preskriptiivisen analyysin tuloksena on ohje yleiselle käyttötapaukselle ottamatta kantaa lopulliseen toteutukseen. Se mahdollistaa ratkaisun kuvauksen jo prosessin suunnitteluvaiheessa. [4, s.133-134.]

Informatiivinen, eli deskriptiivinen analysointi on välivaiheiden toteuttamisen kuvaamista. Deskriptiivisellä analyysilla selvitetään, miten tapahtuma saatiin toteutettua. Esimerkiksi keitettiinkö kahvin keittämiseen käytettävä vesi 100 °C vai alennettiinko veden kiehumispistettä tai käytettiinkö suodatinpaperia vai annettiinko purujen sakkautua pannun pohjalle. Deskriptiivisen analyysin tuloksena on kuvaus yksittäiselle käyttötapaukselle, eli miten haluttuun lopputulokseen päästiin kyseisellä toteutuksella. [4, s.133-134.]

Insinöörityössämme analysoimme prosessin nykytilanteen toimintaa sekä suunnittelemme prosessin, mikä suunnitellaan valmiin ohjailevan analyysin mukaisesti ja kuvatessamme prosessin kehitysvaiheen tulemme analysoimaan samalla kokonaistoteutuksen deskriptiivisesti. Työn lopuksi analysoimme, olisiko ollut mahdollista luoda muita saman liiketoiminnallisen saavuttavia prosesseja ja miten ne olisi tullut toteuttaa.

4 Valvonta

Insinöörityön kohteena on suunnitella prosessi käytettäväksi jatkuvien palveluiden piirissä niin asiakkaalle yksilöidyssä tiimissä kuin kaikki asiakkaat kattavassa jatkuvien palveluiden ylläpitotiimissä. Molempien tiimien päivittäisiin rutiineihin kuuluu lukuisien datasiirtojen, jäljempänä sanomaliikenteen, valvonta. Sanomaliikenne pitää sisällään yhden tai useamman järjestelmän välissä liikkuvat aineistot, jossa yksittäinen sanoma voi olla

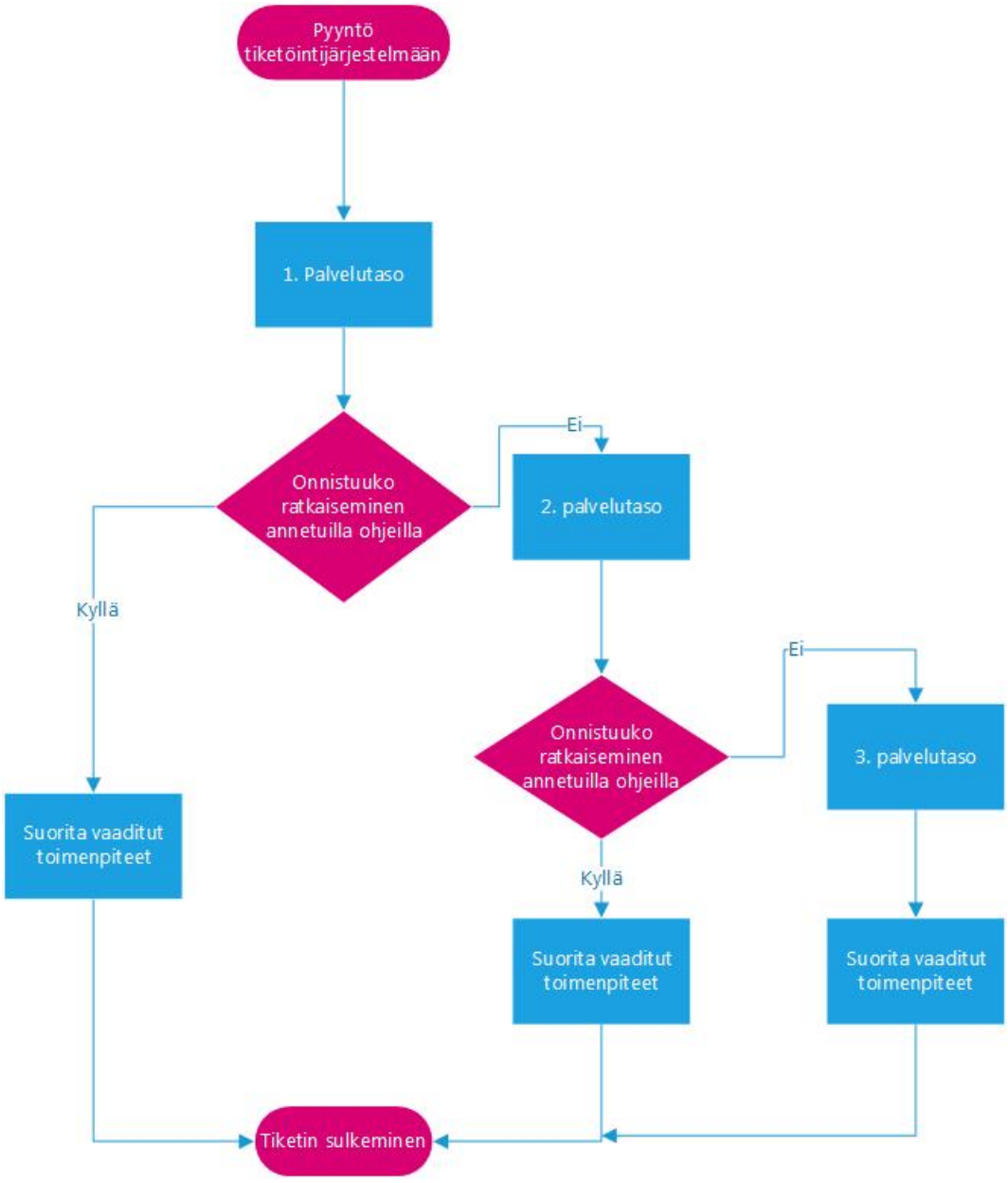
esimerkiksi yksi tiedosto. Sanomalla on aina lähettäjä ja vastaanottaja, ja se voi sisältää dataa tai olla jatkuva informatiivinen heräte, kuten palvelimen vasteaika (ping) tai tiedot lähettäjän palvelimen tilanteesta.

Sanomaliikenteen valvonta ja ylläpitäminen ovat palvelunhallinnan ja integraatioratkaisujen jatko- ja pienkehityksen ohella tärkeimmät jatkuvien palveluiden osa-alueet. Niiden kautta pidetään järjestelmällisesti huolta asiakkaan liiketoiminnalle tärkeästä infrastruktuurista. Yleensä valvontaa ja jatkuvia palveluita tarjotaan eritasoisina vaihtoehtoina palvelusopimuksissa, joissa vaihtelevat valvonnan reagointiajan pituus ja laajuus. Palvelusopimusten muuttujat voidaan räätälöidä asiakkaan tarpeiden mukaisesti, jolloin asiakkaan tuotannolle kriittiset tietovirrat voidaan saattaa nopean reagoinnin alaisiksi ja mahdollisesti asiakkaan sisäiseen tiedonsiirtoon sovellettujen ratkaisujen valvonta voidaan suorittaa niille tarpeellisemmalla reagointivälillä. Valvontaan ja palvelusopimukseen usein sisältyy niin ympäristön terveystarkistukset kuin sanomaliikenteen toiminnan varmistaminen, virhetilaisten viestien käsittely sekä mahdolliset versiopäivitykset eri järjestelmiin. Terveystarkistuksissa seurataan palvelinten ja sovellusten toimintaa, kuten levytilan määrää ja sanomien määrää.

Tiimeille on valmiiksi suunniteltuja toimintaperiaatteita, eli prosesseja koskien kohdattavia ongelmatilanteita. Tietyt ongelmatilanteet, joiden selvittämiseen vaaditaan ympäristön tarkempaa tietämystä, ohjataan asiakasspesifiselle tasolle (kuva 6). Ongelmanselvitys ja häiriötilateiden selvitys toteutetaankin tavallisesti monien eri tasojen avulla, jotta automaattisen ja manuaalisen valvonnan nostamat virhetilanteet voidaan tarpeen vaatiessa siirtää suoraan vaativuuttaan vastaavalle tasolle: tuotetussa palvelussa päivittäin tapahtuvia, niin sanottuja nimellisiä virhetilanteita eskaloidaan ensimmäiselle tasolle usein automaattisen valvonnan kautta. Esimerkkinä on hälytys väärin annetusta käyttäjätunnus-salasana-parista. Integraatioliittymiin kohdistuvia toiminnallisia virhetilanteita, kuten tiedoston uudelleenlähetys, nostetaan integraatioliittymän rakentamisesta vastanneelle taholle tai keskitetylle integraatio- ja jatkuvien palveluiden ylläpidolle ja ohjelmiston tai teknologian perustoimintaan liittyviä virhetilanteita ja ominaisuuksia integraatioteknologian toimittajalle.

Jotta asiakkaan palvelupyyntö löytää nopeasti oikean palveluhaaran on tärkeää järjestää ja ylläpitää kaikki eri tasot tavoittavan palvelunhallintajärjestelmän toiminta asiakaskyselyitä ja asiakkaan parannusehdotuksia varten. Tämä on useimmissa tapauksissa sähköinen tikettijärjestelmä, joka toimii valvonnan ilmoitusten rajapintana joko välillisesti tai

automaation lähettämien viestien muodostamina suorina yksilöitävinä tehtävänantoina: tiketteinä. Tikettijärjestelmä voi toimia viestinnän apuna niin sisäiseen viestintään kuin ulkoisesti ja sen avulla myös ongelmien ratkaisukuvauksia on helppo ylläpitää, sillä jokainen asiakas- tai ongelmatapahtuma, ”incident”, tallentuu historiaan, jotta ratkaisuun voidaan palata vastaavassa tilanteessa myöhemmin. Yksittäisestä ratkaisutilanteesta jäävät talteen myös tarvittavat viitemerkinnät ja statistiikat.



Kuva 6. Prosessikaavio palvelupyynnön käsittelystä

Järjestelmässä kohdatut ongelmat on myös hyvä koota palvelutietämyksen hallintajärjestelmän (service knowledge management system) tietokantaan, johon hyväksi havaitut

selvitystoimenpiteet voidaan listata ongelman seuraavaa kohtaajaa varten. Palvelutietämyksen hallintajärjestelmässä säilytetään tietämyksen, informaation ja tiedonhallintaan käytettäviä työkaluja ja tietokantoja, ja se kattaa tiedon jokaisen jatkuvien palveluiden käytössä olevista käytännöistä. Myös mahdolliset virhealttiit osat ja mahdollinen tiketin siirtäminen omalle vaativuustasolleen tiettyjen tunnusmerkkien täyttyessä merkitään hallintajärjestelmään ongelman nopeaa selvittämistä varten.

4.1 Valvontaprosessit

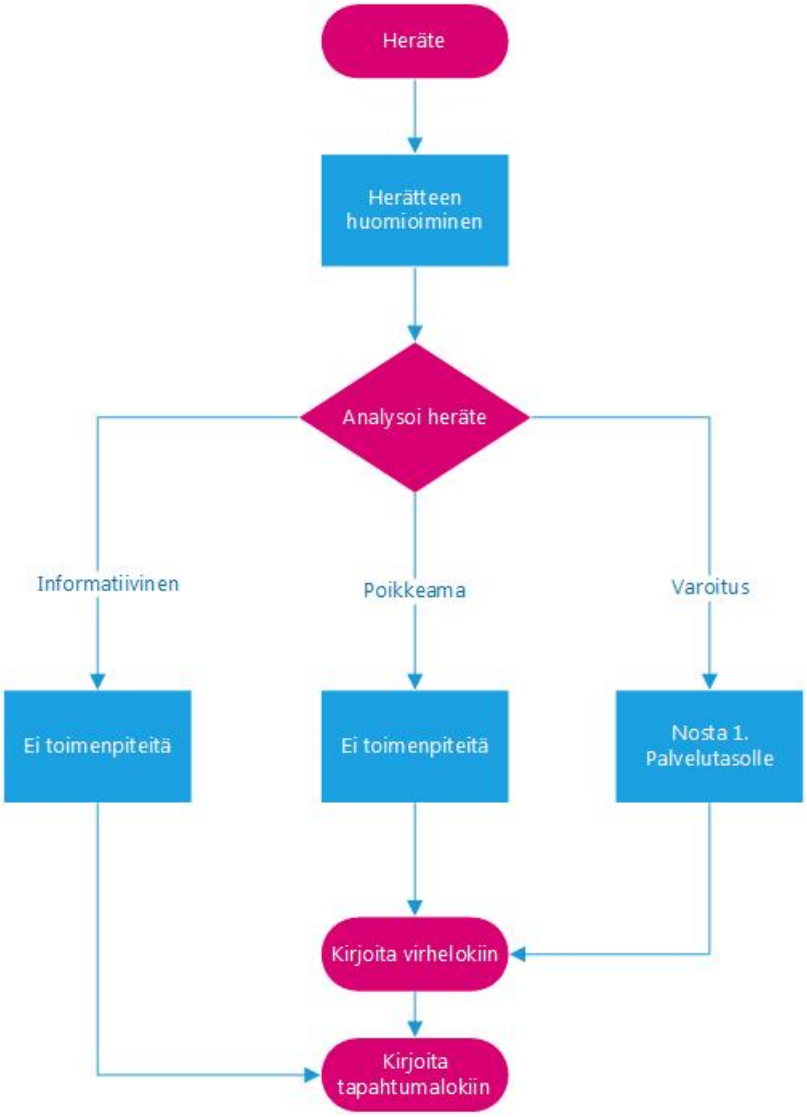
Prosessi monitoroidulle lokiseurannalle tarkoittaa prosessia jatkuvien palveluiden valvontaprosessien visuaaliselle hyödyntämiselle häiriönhallinnassa ja ongelmanratkaisussa. Prosessin avulla valvontaprosessien nostamat herätteet ja häiriöt saadaan tuotettua tarpeen vaatimalle tasolle mahdollisimman yksinkertaisella ja yksiselitteisellä tavalla. Tällä tavoin nopeutetaan reagointiaikaa herätteeseen ja pienennetään liittymän palvelupolkua.

Kullekin palvelutasolle sovelletaan tasolle sopivia valvontaprosesseja. Palvelun laadulle on epäedullista palvelutasojen tarpeeton kuormittaminen ja usein tasoille jalkautetut virheenhallinnan prosessit ja -käytännöt eroavat toisistaan. Onkin tärkeää, että oikea tilanne kohtaa oikean palvelijan oikeaan aikaan. Eriävät prosessit optimoivat parhaimmillaan tilanteeseen reagointia, mutta niihin yhdistettynä vääränlaiset herätteet ovat tiimin toiminnalle epäedullisia ja voivat pahimmassa tapauksessa sotkea perustettujen prosessien toiminnan. Esimerkiksi kehitystiimille tärkeämpiä ovat projektin etenemisestä saadut informatiiviset herätteet kuin palvelimen automaattisesti lähettämät, poikkeaman osoittavat herätteet.

4.1.1 Herätteidenhallinta (event management)

Heräte (event) on mikä tahansa tilamuutos, jolla on merkitystä konfiguraation tai IT-palvelun hallinnalle. Se käsittää myös tilamuutoksia, jotka eivät välttämättä ole IT-palvelun käyttäjälle itsestäänselvyyksiä: sisäänkirjautuminen on tilamuutos, jossa uloskirjautunut käyttäjä vaihtaa tilansa sisäänkirjautuneeksi. Jokaisesta tilamuutoksesta jää merkintä palvelulokeihin, joten jokainen toiminta on yksilöitävissä. Herätteidenhallinnan päämääränä on havaita ja selvittää tarpeelliset herätteet ja seuloa tarpeelliset massasta, luoda perusta operatiiviselle seurannalle ja valvonnalle sekä toimia alustana palvelutuotannon

rutiininomaisten tehtävien automatisoinnille (kuva 7). Herätteillä suoritettava valvonta voi olla joko automatisoitua tai manuaalista ja herätteet jaotellaan informatiivisiin herätteisiin, poikkeamiin ja varoituksiin. [4, s.96-97.]



Kuva 7. Esimerkki herätteiden käsittelystä. Informatiivinen, poikkeama ja varoitus. [5, mukaillen s.95]

Informatiivinen heräte (informational) on merkityksellinen liittymän normaalitoiminnalle. Se on ilmoitus aikataulutetun työn valmistumisesta, sähköpostin saapumisesta vastaanottajalle tai muusta tapahtumasta, jossa liittymää tai ohjelmistoa käytetään tarkoituksenmukaisella tavalla. Jokaisesta sanomaliikenteen eri vaiheesta jää merkintä palvelimelle

lokeihin, joten jokainen näistä voidaan nähdä informatiivisena herätteenä (kuva 8). Kaikkia ei kuitenkaan ole tarpeellista eikä tarkoituksenmukaista raportoida eteenpäin tiimille, sillä jokainen palvelimen kosketus ulkomaailmaan, jokainen sisään- ja uloskirjautuminen ynnä muut aiheuttaisivat raporttiin loputtoman informaatiotulvan, jolloin herätteiden käyttäminen menettäisi tarkoituksensa. Informatiivisilla herätteillä voidaan kuitenkin varmistaa tiimien oikea resursointi, sillä niiden oikeanlaisilla käytöllä on mahdollista muodostaa nollailmoituksia, joiden avulla tiimi voi yhdellä vilkaisulla varmistua asiakkaan ympäristön olevan ongelmattomassa tilassa. [5, s.96-97.]

```
14:09:11,673 [Thread-59] DEBUG - Setting entrypoint for message M8B4E4IO05IRD.
14:09:11,673 [Thread-59] DEBUG - Writing not silent message object into database with message ID: M8B4E4IO05IRD
14:09:11,685 [Thread-59] DEBUG - Message object with ID: M8B4E4IO05IRD was successfully created.
14:09:11,692 [Thread-51] DEBUG - Status PROCESSING was set for message M8B4E4IO05IRD. New locking status for message is 0
14:09:11,701 [Thread-51] DEBUG - Setting sender node for message M8B4E4IO05IRD.
14:09:11,702 [Thread-51] DEBUG - Setting message type into message M8B4E4IO05IRD.
14:09:11,707 [Thread-51] DEBUG - Update msg=M8B4E4IO05IRD data to 08B4E4IO05IRE.
14:09:11,716 [Thread-51] DEBUG - Setting receiver node for message M8B4E4IO05IRD.
14:09:11,716 [Thread-51] DEBUG - Same datas, close for association
```

Kuva 8. Esimerkki integraatiosovelluksen sanomakäsittelyn informatiivisten herätteiden lokikirjauksista.

Poikkeama (exception) on heräte tilanteesta, jossa liittymää, ohjelmaa tai ympäristöä käytetään käyttötarkoitukseen kuulumattomalla tavalla, kuitenkaan sen toimintaa estämättä. Poikkeaman voi aiheuttaa loppukäyttäjän kirjautumisyritys ohjelmistoon väärällä salasanalla, laitteistolle aiheutunut normaalia suurempi kuormitus, liiketoimintaprosessissa ilmennyt epätavallinen tilanne tai ympäristössä tapahtuva ylimääräinen tapahtuma, esimerkiksi mahdollinen haittaohjelma. Poikkeamaherätteiden avulla saatetaan palveluntarjoajan tietoon mahdollinen epäasiallinen toiminta ympäristössä. [5, s.96-97.]

Palvelun kohtaama epätavallinen, mutta normaaliin toimintaan kuuluva tilanne nostaa varoitusherätteen (warning). Varoitus syntyy muun muassa tilanteesta, jossa liittymän toteutusajat ovat hetkellisesti kasvaneet tai palvelimen käyttöasteet lähestyvät suurinta sallittua raja-arvoa. Tämä voi johtua esimerkiksi hetkellisestä, normaalia suuremmasta palvelun kysynnästä, mikä ilmenee kasvuna palveluun otettujen yhteydenottojen määrässä, mikä johtaa yksittäisen yhteyden palveluajan kasvuun ja jonka seurauksena yksittäinen istunto palvelussa kestää keskimääräistä pidempään. Istunto varaa resursseja saman määrän kuin normaalisti, mutta ne vapautuvat normaalia pidemmän ajan kuluttua, jolloin palveluun on mahdollisesti ehditty ottaa yhä useampia ja useampia yhteyksiä lisää. [5, s.96-97]

Herätteidenhallinnan tavoitteena on havaita kaikki tarpeelliset tilamuutokset, päättää kullekin herätteelle sopiva toimintamalli ja -toimenpide sekä varmistaa näiden toimenpiteiden kommunikointi eteenpäin. Valvomalla herätteitä tiimi pysyy selvillä palvelinten kuormituksesta ja mahdollisista vikatilanteista, joihin tiimin tulee reagoida palvelusopimuksen mukaisesti. Herätteiden avulla tarjotaan lähtökohta palvelutuotannon prosesseille ja käyttöpalvelun hallinnan toiminnolle sekä tarjotaan keinot vertailla operatiivista suorituskykyä ja toiminnan kehittymistä. Prosessien toiminta vaatii raaka-aineeksi herätteitä ja käyttöpalvelusta saadaan palautetta oikeanlaisten toimintamerkintöjen kautta. Palvelu voi olla olemassa pitkäänkin, mutta ilman palautetta ei voida varmistua sen oikeasta toiminnasta eikä siitä, että palvelua edes käytetään.

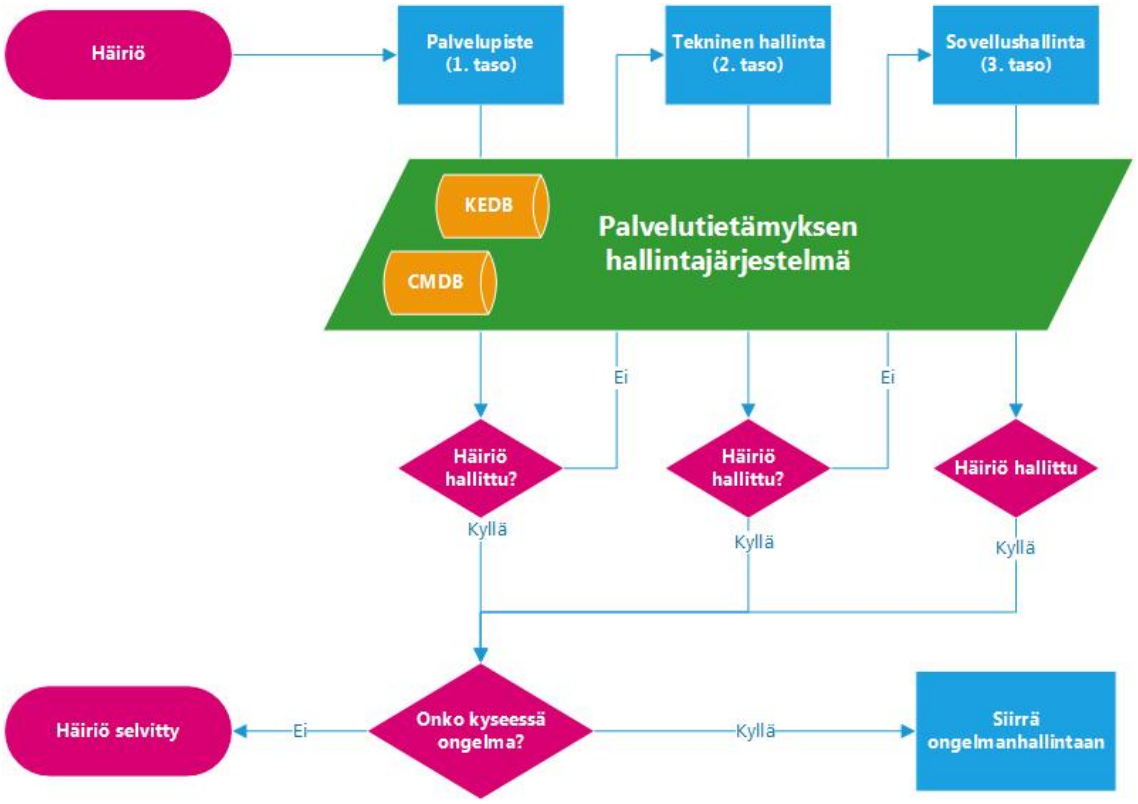
4.1.2 Häiriönhallinta (incident management)

Häiriö (incident) on datasiirron suunnittelematon keskeytys, IT-palvelun rakenneosan vikautuminen tai palvelun laatua laskeva tila integraatioliittymässä. Nopeaan häiriönhallintaan kuuluu tilapäisratkaisu (workaround), mikäli ongelman lopulliseen ratkaisuun kuluvana aikana liittymä ehtisi olla poissa käytöstä. Häiriö muuttaa aina palvelun toimintaa aiheuttaen mahdollisesi myös keskeytyksen. Häiriön tutkinta sekä selvitystyö ovat ensisijaisen tärkeitä, kunnes vähintään tilapäisratkaisu on saatavilla. [5, s.96-97]

Tilapäisratkaisulla häiriön tai ongelman vaikutusta pyritään vähentämään tai optimaalisesti poistamaan kokonaan (kuva 9). Tarkoituksena on kuitenkin palauttaa palvelu aina sille tasolle, että sen käyttöä pystytään jatkamaan. Prosessin määrittelemä manuaalinen toimenpide, esimerkiksi asiakasrajapinnan mahdollistaman virheellisen käyttötapauksen takia, voi ajoittain muistuttaa tilapäisratkaisua. Kyseessä ei kuitenkaan ole itse järjestelmän häiriö; häiriö on aina laajempi kokonaisuus ja aiheuttaa kokonaisten liittymien virhetilanteita. [5, s.96-97]

"Tilapäisestä ratkaisusta muodostuu usein pysyvä ongelma."

– Craig S. Bruce, 1986 –



Kuva 9. Esimerkki häiriön käsittelystä ja ohjaamisesta oikealle palvelutasolle tai häiriöön toistuessa ongelmanhallintaan. Kuvan yläreunan CMDB (configuration management database) ja KEDB (Known Error Database) ovat osa palvelutietämyksen hallintajärjestelmää. (mukaillen kuvaa KEDB)

Kun kyseessä on toistuvasti ilmaantuva häiriö, on mahdollisesti prosessin suunnittelussa tapahtunut virhe tai kaikkia prosessiin vaikuttavia muuttujia ei ole otettu huomioon. Tällöin on kyse ongelmanhallinnasta, jolloin usein prosessi otetaan mukaan tarkempaan prosessianalyysiin tai prosessin kehitys otetaan uudelleen työn alle.

Hälytys (alert) on ilmoitus, että jonkin mitattavan määreen kynnysarvo on saavutettu tai palvelu on vikaantunut. Hälytykseen reagoinnin laiminlyönti johtaa aina palvelun toiminnan keskeytymiseen ja hälytyksen tarkoituksena on nostaa esiin pian käsillä oleva palvelun katastrofitilanne. Esimerkki katastrofitason hälytyksestä on palvelimelle allokoidun tallennustilan loppuminen kesken. Välttämättä tallennustilaa ei ole vielä käytetty kaikkea, mutta palvelun kriittisiin toimenpiteisiin tarvittavaa määrää ei ole enää jäljellä.

Ongelmatilanteen realisoituessa häiriönhallinta palauttaa palvelutuotannon kyvyn toimia, ja tarkoituksena on minimoida ongelmatilanteen vaikutus liiketoimintaan. Häiriönhallinnalla varmistetaan standardoitujen menetelmien ja toimintatapojen käyttäminen häiriöti-

lanteiden käsittelyssä sekä parannetaan häiriötilanteen läpinäkyvyyttä koko organisaation sisällä. Tarkoituksena on ylläpitää käyttäjätyytyväisyyttä, sillä tuotetun palvelun tulee olla myös käytettävissä. [5, s.101]

4.1.3 Dataliikenteen analyysi

Dataliikenteen analyysi on erilaisten mittarien ja raja-arvojen hyödyntämistä sanomavirran läpi kulkevan dataliikenteen mittaamiseksi. Dataliikenteestä analysoidaan usein sanoma- tai datamääriä tietyn ajanjakson sisällä, sanomaliikenteen suorituskykyä ja mahdollisten häiriötilanteiden määrää ja laatua. Analysoinnin kautta voidaan nostaa esiin erilaisia prosessien epäkohtia ja saadaan kerättyä arvokasta tietoa niin yrityksen liiketoiminnalle kuin asiakkaan tärkeiden prosessien kehitykseen.

Dataliikenteen sanoma- ja datamäärien mittaamista tietyllä aikavälillä käytetään hyödyksi infrastruktuurin eri ohjelmistojen resurssien arviointiin ja määrittämiseen. Tämä liittyy samalla hyvin läheisesti suorituskyvyn mittaamiseen. Sanomien käsittelyajan monitorointi niin sanomaliikenteen pienemmissä komponenteissa kuin kokonaisuudessa on hyvin tärkeässä osassa varsinkin reaaliaikaisten ja lähes reaaliaikaisten liittymien toiminnassa. Mahdolliset toimenpiteet dataliikenteen optimoimiseksi parantavat huomattavasti sovelluksen toimintaa loppukäyttäjän näkökulmasta. [6, s.146-147]

Häiriötilanteiden määrän ja laadun analysointi on yksi tärkeimmistä edellä mainituista mittareista. Häiriötilanteiden kehittyneellä analysoinnilla mahdollistetaan integraatioliittymien jatkuva kehittäminen. Virheiden määrä eri kokonaisuuksissa voi kieliä mahdollisista kriittisistä virheitilanteista, ja näiden tilanteiden tarkka analysointi, virheen aiheuttajan selvitys ja dokumentointi auttavat ennaltaehkäisemään tulevia virhetilanteita. Virheiden luokittelu tasojen laajuuden ja kriittisyyden mukaan auttaa valvontatyötä tekevää työntekijää tunnistamaan mahdollisen kriittisyyden tason nopeammin ja oikean palveluhaaran selvittäminen ja virheestä toipuminen nopeutuu huomattavasti. [6, s.148.]

Kaikkien osa-alueiden osalta on erittäin tärkeää kuitenkin ennalta määritellä mitattavat määreet. Hyvin suunnitellulla dataliikenteen analyysillä saadaan dataliikenteen toiminnasta sujuvampaa sekä parannettua vikasietoisuutta. Häiriönhallinnassa käytetään usein avuksi tarkistuslistoja, joiden avulla suoritetaan toimenpiteitä tietyssä järjestyksessä ja pidetään sidosryhmät ajan tasalla tilanteesta. Esimerkiksi laajan häiriötilanteen selvittämiseen on ennalta määritellyt toimenpiteet: selvitetään ongelmakohta, lähetetään

sidosryhmille häiriöilmoitus, selvitetään ongelman juurisyy, korjataan ongelma ja ilmoitetaan häiriötilanteen päättymisestä sidosryhmille.

4.2 Valvontatyökalut

Valvontatyökalut voivat olla joko passiivisia tai aktiivisia. Passiivinen valvontatyökalu ei suorita itse muuta toimintaa kuin tiedon koostamista. Se koostaa tietoa esimerkiksi sanomavirran tilasta ja sen läpi kulkeneiden sanomien lähetysstatuksesta. Passiivisena toimii muun muassa sanomavirran visuaalinen monitorointityökalu, josta voi nähdä viimeisimmät onnistuneet ja epäonnistuneet sanomat. Lisäksi se voi tarjota lisätietoa virheen laadusta ja aiheuttajasta. Aktiivinen valvontatyökalu puolestaan tekee itse jotain valvontainformaation luomiseksi. Esimerkiksi palvelimen jatkuva viive- ja saatavuuskysely on aktiivista valvontaa kuin myös jokainen palvelua vasten rakennettu mock-testirakenne. Mock-testirakenteella tarkoitetaan järjestelmän peilikuvaa, joka sisältää vain testitapauksille tarpeelliset palautukset todellisen järjestelmän toiminnallisuuksista. [7, s.150-151.]

Aktiivisena valvontatyökaluna voidaan nähdä myös valvontatyötä tekevä valvontatiimin työntekijä tai valvontatiimin päivittäiseen toimintaan sisällytetty valvontaprosessi. Päivittäiset aamutarkistukset ja joka viikko suoritettavat terveystarkistukset ovat hyödyllisiä valvontatyökaluja integraatiototeutuksen kokonaisvalvonnalle.

Integraatioalustan käyttöliittymä on olennaisessa osassa datasiirtojen valvonnassa. Käyttöliittymän avulla saadaan helposti koostettua kokonaiskuva siirtojen tilasta. Alustan avulla voidaan esimerkiksi hakea tietyn päivän tietyn statuksen omaavat sanomat (kuva 10) ja nähdä viimeisimmät toteutuneet virhetilanteet. Verraten muihin valvontatyökaluihin integraatioalustan käyttöliittymä antaa huomattavasti tarkemmin tietoa sanomien tilasta ja mahdollisten virhetilanteiden syystä.

Integraatioalustan käyttöliittymän käyttäminen monitorointityökaluna saattaa kuitenkin sisältää haasteita. Käyttöliittymä saattaa antaa virheellistä informaatiota, mikäli sitä ei ylläpidetä aktiivisesti. Osa sanomavirroista on toteutukseltaan sen kaltaisia, että niiden päätyessä virheeseen ei virheellistä aineistoa voida käsitellä, vaan se on lähetettävä uudelleen lähettävästä päästä. Tällöin on mahdollista, että käyttöliittymälle jää käsittelemättömiä virhesanomia. Aiheettomat virhe-statuksella olevat viestit taas sekoittavat

mahdollisia raportteja, joita ajetaan integraatioalustan käyttöliittymältä, mikäli käyttöliittymää ei pidetä ajan tasalla.

Message count	Status
13199	SENT
420	ERROR
401	FAIL

Kuva 10. Esimerkki integraatioalustan raportista

Integraatioalustan käyttöliittymältä on helppo hallinnoida toteutettuja sanomavirtoja ja nähdään reaaliajassa, mitkä sanomavirrat ovat aktiivisia ja mitkä on asetettu pitoon. Käyttöliittymältä voidaan asettaa sanomiin tehtäviä muunnoksia, eli transformaatioita, joilla sisään luettu sanoma saadaan muutettua toiseen muotoon tai sen sisältämää tietoa voidaan käyttää hyödyksi niin loogisesti kuin uuden aineiston luomiseen.

4.3 Valvonnan haasteet

Ennalta määrittelemättömät ja liiketoiminnalle kriittiset, pikaista selvitystä vaativat virhetilanteet ovat muutamia esimerkkejä valvontaprosessien haasteista. Valvontaprosessien toiminta ylemmillä palvelutasoilla perustuu pitkälti vanhojen virhetilanteiden dokumentointiin ja niiden avulla kartoitettuun osaamiseen. Jokaisesta virhetilanteesta jää merkintä palvelimelle herätteiden kautta ja jokaisesta valvontahenkilöstön suorittamasta ongelmanselvityksestä tulisi jäädä merkintä ja ratkaisukuvaus palvelutietämyksen hallintajärjestelmään. Tällöin ongelmatilanteen toistuessa ratkaisukuvaus on valmiiksi saatavilla eikä ratkaisun selvittämiseen tai tilanteen analysointiin tarvitse käyttää uudestaan samaa määrää resursseja. Hallintajärjestelmänä toimii usein yhtiön sisäinen keskitetty informaatiokeskus, joka on kaikkien organisaation työntekijöiden muokattavissa ja jonne voidaan perustaa myös tietoturvallisia, rajoitettuja kokonaisuuksia. Toisin sanoen hallintajärjestelmä on kuin sisäinen wiki.

Valvonnalle tulee määrittää prosessit annettujen työkalujen päivittäiseen käyttöön ja ohjeet siitä, miten työkalujen tuottamaan dataan tulee reagoida. Ilman rutinoituneita prosesseja on vaarana, että valvontaa suoritetaan harvakseltaan ja satunnaisesti. Mitä enemmän valvottavia asiakkaiden yhteyksiä alistetaan tiimin valvontaan, sitä suurempi vaara on laiminlyödä yhteyksien valvonta. Valvonnan laiminlyönti voi johtua esimerkiksi resursoinnin epätasaisuudesta, jolloin vastuuhenkilölle on kasautunut samalta tai usealta asiakkaalta tärkeysjärjestyksessä valvonnan ohittamia virheenselvitys- ja palvelupyyntöjä. Asiakkuuden valvonnan vastuuhenkilöllä on myös vastuu ohjata työtä eteenpäin, mutta on myös mahdollista, ettei vapaata työskentelykaistaa löydy tiimin sisältä. Tällöin valvontatyötä suorittavan tiimin toiminta on syytä resursoida uudelleen ja kartoittaa lisätyövoimaa tiimin ulkopuolelta. Tästä syystä ajan tasalla oleva palvelutietämyksen hallintajärjestelmä on elintärkeä jaettaessa vastuuta tiimin ulkopuolelle ja otettaessa tiimiin sisään uusia jäseniä.

Valvonnassa hyväksi käytettävien herätteiden oikeanlainen hyötykäyttö on tärkeä osa tarkoituksenmukaisten prosessien kehittämistä. Herätteiden määrään panostaminen ei välttämättä helpota valvontatyön suorittajan työtä, mikäli iso osa herätteistä on turhanpäiväisiä. Nostettujen herätteiden tulisi olla kriittisyystasoonsa suhteutettuina. On olemassa kuitenkin informatiivisia herätteitä, jotka ovat yhtä tärkeitä kuin häiriöt ja hälytykset. Esimerkiksi pääkäyttäjätunnuksilla sisäänkirjautumisen tulisi nostaa kriittinen informatiivinen heräte, jotta palvelun tietoturvaa ja sen tasoa on mahdollista valvoa ja väärinkäyttöön mahdollista puuttua reaaliaikaisesti.

Valvottavia ympäristöjä on mahdollista olla yhtä monta erilaista kuin yrityksellä on asiakkaita. Asiakkaiden käyttämät teknologiat jo esimerkiksi yhteyden muodostukseen vaihtelevat sopimuksittain, ja muuttujat asiakkaiden välillä riippuvat usein asiakkaan sidosryhmistä. Verkkoliikenteen toteutus on voitu toteuttaa asiakkaan sisäverkkoon, jolloin yhteydenotto tulee suorittaa erillisen VPN-sovelluksen kautta (virtual private network) tai verfioida yhteys palvelimelle muulla tapaa, esimerkiksi kaksisuuntaisesti erillisen nettisivun ja mobiilivarmenteen kautta. Asiakkaalle on voitu myydä myös täyden palvelun sopimus, jolloin kaikki tuotettu löytyy tuottajan omasta sisäverkosta, eikä yhteyksiä sidosryhmiin tällöin välttämättä tarvita. Ympäristöissä käytetyt ja yhteyksiä toteuttavat ohjelmistot voivat olla useiden eri ohjelmistotuottajien tuottamia ja kuulua eri tuoteperheisiin, joten valvontatyö vaatii useisiin eri teknologioihin perehtymistä. Esimerkkeinä näistä Microsoftin BizTalk Server sekä IBM WepSphere -tuoteperheet. Lisäksi on olemassa monia muita ei-kaupallisessa käytössä olevia tuotteita. Valvontaan kouluttautuminen vaatii

vasta-alkajalle perehtymisjakson ja kouluttajan, mikä tulee ottaa huomioon resursoinnissa.

5 Nykytila ja tavoitteet

Insinöörityön kohteena olevan IBM WebSphere -tuoteperheen jonomanagerin - joka on sanomapohjainen väliohjelmisto - virhetilanteiden analysoinnin perustana oleva integraatiototeutus ja sen kirjoittama lokidata kuuluvat nykytilaan. Jonomanagerien lävitse kulkee sanomia päivittäin ja niistä kirjautuu aktiivisesti lokikirjauksia.

5.1 Analysointi

Nykytilassa jonomanagereiden tuottamaa lokidataa ei analysoida, vaan siitä kerätään pelkkä virhetilanteiden kokonaismäärä. Virheiden kokonaismäärästä voidaan päätellä vain, toimiiko integraatiototeutus moitteettomasti vai ei. Siitä ei ole apua virhetilanteiden proaktiiviselle ratkaisulle, sillä tyypistä riippumatta virhetilanne luetaan mukaan laskuriin.

5.2 Nykytila

Jatkuvat palvelut on integraatioiden kannalta olennainen palvelu asiakkaalle. Sen avulla asiakkaan integraatiot pysyvät ajan tasalla, ja mahdollisten virhetilanteiden selvittäminen tapahtuu hallitusti. Jatkuvien palvelujen tehtävänä on ylläpitää asiakkuuden häiriö- ja ongelmahallintaa sekä parantaa asiakkaalle tarjottuja palveluita ja niitä koskevia tai erillisiä prosesseja. Jatkuvat palvelut on jaettu kolmeen tasoon ITIL:in mukaisesti: Palvelupisteeseen (Service Desk), tekniseen hallintaan ja sovellushallintaan.

- Palvelupiste (1. taso) toimii 24 tuntia vuorokaudessa, seitsemän päivää viikossa. Palvelupisteen tehtävänä on ratkaista yksinkertaisimmat etukäteen ohjeistetut hälytykset, eskaloida käyttäjäilmoitukset seuraavalle tasolle ja monitoroida herätteitä. Palvelupisteen toimenkuva sisältää yksinkertaisten häiriöilmoitusten vastaanottamisen puhelimitse tai häiriöilmoituksin.
- Teknisen hallinnan (2. taso) tehtävänä on selvittää ja ratkaista palvelupisteen eskaloimat häiriöilmoitukset. Sen tehtäviin kuuluu myös yksinkertaisten pienkehitysten tekeminen. Teknisessä hallinnassa seurataan ohjelmiston eri komponenttien toiminnallisuutta ja ennaltaehkäistään mahdollisia virhetilanteita.

- Jatkuvien palvelujen kolmannen tason, eli sovellushallinnan kehitystiimin pääasiallisena tehtävänä on kehittää jatkuvien palvelujen alaisuuteen tarjottuja tuotteita ja palveluita. Kehittäjät toimivat myös teknisen hallinnan tukena sekä 1. ja 2. tasojen ohjeistajana.

5.3 Tavoite

Insinöörityössä kehitetään jatkuvien palvelujen toiselle tasolle, tekniseen hallintaan työkalu ja prosessi, jonka avulla mahdollistetaan jonomanagerien virhetilanteiden tunnistaminen ja ennaltaehkäisy.

Prosessin suunnittelusta haasteellisen tekee se, että mahdollisten virhetilanteiden tunnistaminen ja proaktiivinen ratkaiseminen prosessin elinkaaren alkuvaiheessa on hyvin vaikeaa ennakkotapausten puuttumisen vuoksi. Tästä syystä prosessin kehittämisessä on otettava huomioon se, miten ja minne virhetilanteista raportoidaan ja kerätään niin sanottua virhehallinnan tietokantaa keskitettyyn palvelutietämyksen hallintajärjestelmään. Jatkuvan tiedonkeruun avulla prosessista saadaan mahdollisimman tehokas ja kykenevä mukautumaan uusiin virhetilanteisiin sekä itse prosessin kehittymiseen. Proaktiivisella häiriöiden ja ongelmatilanteiden ratkaisulla estetään virheiden kasautumisen aiheuttavien liiketoimintaa estävien virhetilanteiden realisoituminen.

Prosessin tulee tarjota monitoroitu näkymä jonomanagereiden läpi kulkevien sanomavirtojen kirjoittamaan lokidataan, josta on suodatettu läpi ongelmanratkaisussa tarpeellinen. Näkymän tulee olla helpposelkoinen ja siitä tulee olla helposti selvitettävissä mahdollisen ongelmatilanteen ratkaisuun tarvittavat lisätiedot.

5.4 Käytetyt työkalut

Jatkuvien palveluiden valvonta- ja ratkaisutukitiimi valvoo useiden eri asiakkaiden integraatiototeutuksien sanomavirtoja, joiden toteutus eriää toisistaan. Insinöörityössä suunniteltavan prosessin työkaluihin kuuluvat integraatiototeutuksen toteuttavat osat, valvontaa suorittavat työkalut, toteutuksen sanomaliikenteestä lokidatan kirjoittavat työkalut ja kirjoitettua lokidataa analysoivat ja läpikäyvät työkalut, analysoidun datan visualisoivat työkalut sekä palvelutietämyksen hallintajärjestelmän eri rooleissa toimivat työkalut.

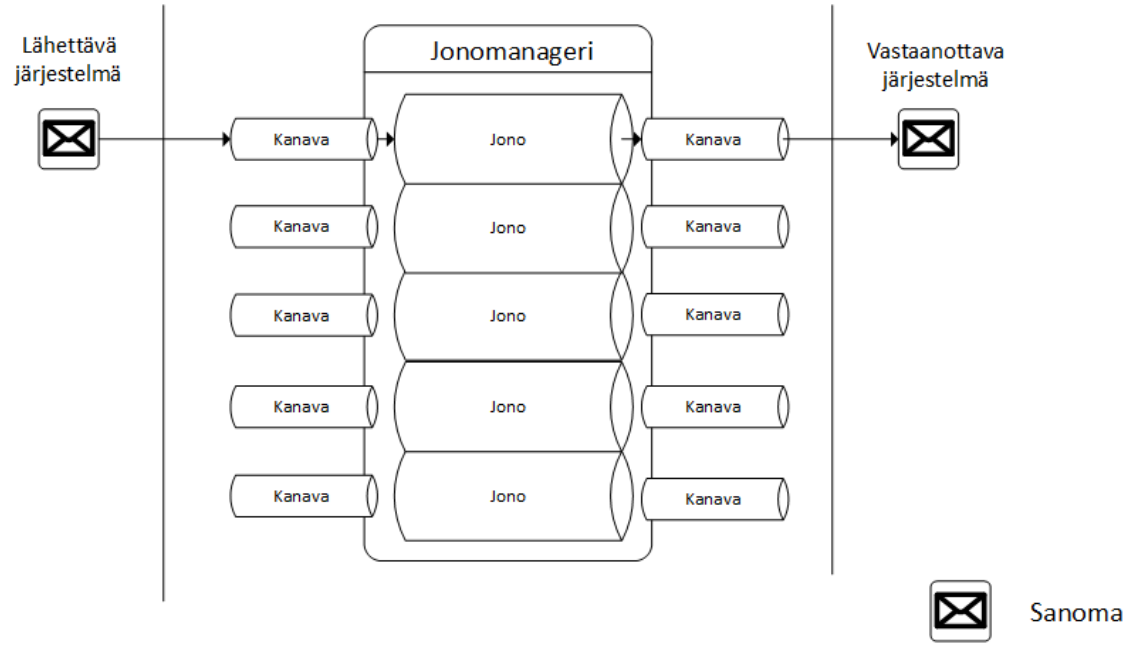
5.4.1 IBM MQ

IBM MQ on sanomavälityksen väliohjelmisto. Se yksinkertaistaa ja nopeuttaa sovellusten integroimista toisiinsa ottamatta kantaa siihen, missä tai millä alustalla sovellukset toimivat. Datan, eli sanomien, välitys tapahtuu jonomanagereissa kanavien ja jonojen kautta. Yksittäiseen jonoon kasautuu sanomia, joita voidaan lukea jonosta asynkronisesti, jolloin vastaanottava järjestelmä pystyy hakemaan sanomat silloin, kun se on valmis vastaanottamaan ne.

Jonomanagerit ovat ylätason objekteja IBM MQ:lla. Ne pitävät sisällään kaiken sanomanvälitykseen liittyvän, kuten jonot ja kanavat. Jonomanagerin tehtävänä on käynnistää kanavat, prosessoida client-sovellusten lähettämiä kutsuja sekä luoda, poistaa ja määrittää jonojen ja kanavien määrityksiä (kuva 11).

Jonot mahdollistavat hallitun ja kronologisen sanomavälityksen IBM MQ -tuotteessa. Jonoissa säilytetään sanomat oikeassa järjestyksessä niin kauan, kunnes ne käydään lukemassa vastaanottavan pään toimesta. Jonoille määritellään suurin sallittu jonosyvyys ja suurin sallittu sanoman koko.

Kanavat ovat olennainen osa MQ:n toiminallisuutta. Kanavia on kahta eri tyyppiä: sanomakanava, jonka avulla sanomia välitetään jonomanagerien välillä ja MQI-kanava, jonka avulla yhteydessä olevat sovellukset suorittavat komentoja sanomien kirjoittamiseen ja lukemiseen liittyen. [8.]



Kuva 11. Yksittäisen jonomanagerin kautta kulkeva sanomaliikenne

5.4.2 Logstash

Logstash on Elastic co. hallinnoima avoimen lähdekoodin työkalu datankeräykseen. Logstash-työkalun avulla voidaan dynaamisesti yhdistellä dataa eri tietolähteistä ja normalisoida halutuin kriteerein. Pääasiassa kyseessä on lokipohjainen datakeräin, ja se on tarkoitettu analytiikan, visualisoinnin, monitoroinnin, arkistoinnin ja virhetilanteiden hallitsemisen käyttötarkoituksiin.

Seuraavassa kuvassa (kuva 12) esitetään Logstash-clientin logstash.conf-tiedostoa, jolla määritellään kyseisen clientin lokitiedostojen sisäänluku ja parsinta ja edelleenlähetys. Kuvassa ylin elementti ”input” kuvaa lokitiedoston sisäänlukua. Tässä tapauksessa on määritelty, että tiedostoa aloitetaan lukemaan alusta ja tiedoston tyyppi on lokitiedosto. Elementin ”filter” avulla sisään luetusta tiedostosta saadaan poimittua tarvittavat tiedot haluttuun tietorakenteeseen jatkokäsittelyn helpottamiseksi. Esimerkissä mainitun viimeisen elementin ”output” avulla poimitut tiedot saadaan lähetettyä eteenpäin haluttua protokollaa käyttäen. [9.]

```
input {
    file {
        path => "/home/user/nginxAccess.log"
        start_position => "beginning"
        type => "logs"
    }
}
filter {
    grok{
        match=>{
            "message"=>"%{IP:clientip} \- \- \[%{NOTSPACE:date} \-%{INT}\] \"
            %{WORD:action} /%{WORD}/%{WORD}/%{NOTSPACE:login} %{WORD:protocol}/%{NUMBER:protocolNum}\"
            %{NUMBER:status} %{NUMBER} \"%{NOTSPACE}\" \"%{NOTSPACE:client} \(%{WORD};
            %{WORD:clientOs}%{GREEDYDATA}"
        }
        add_field=>{
            "eventName"=>"grok"
        }
    }
    geoip {
        source => "clientip"
    }
}
output {
    elasticsearch {
        protocol => "http"
        host => "QBOX_ES_IP:ES_PORT"
        index => "logstash-test-01"
    }
}
```

Kuva 12. Esimerkki logstash.conf-tiedostosta [10.]

5.4.3 QlikView

QlikView oli olennainen työkalu prosessin tavoitteiden kannalta, koska QlikView-näkymälle rakennettu analytiikka toimii prosessin seurantatyökaluna. Tarvitsimme ohjelmiston tuottaman näkymän, jonka avulla muuttuvan tilanteen valvominen kävisi helposti ja yksinkertaisesti. Tarkoituksena oli luoda erillinen fyysinen näkymä, joka olisi jatkuvasti tiimin saatavilla. QlikView on Qlik-yrityksen kehittämä ohjelmistojärjestelmä, jonka avulla pystytään rakentamaan erilaisia näkymiä kunkin käyttäjän omiin tarpeisiin soveltuviksi (kuva 13).

Sovelluksen avulla voidaan visualisoida haluttua esikäsiteltyä dataa monessa muodossa ja eritellä aineistojen välisiä suhteita. Esimerkiksi tämän opinnäytetyön kohdalla ohjelmistoa käytetään rakentamaan lokidatasta erilaisia havainnollistavia näkymiä hyviksi todettujen kriteerien avulla. QlikView-sovelluksen avulla datan analysointi on reaaliaikaista ja nopeaa. QV-näkymällä voidaan lisäksi asettaa haku- ja suodatusvaihtoehtoja, joiden avulla käyttäjä pystyy kohdentamaan näkymän tarvittaessa suppeammalle tarkastelualueelle ja näin tarkastelemaan myös laajempia kokonaisuuksia yksityiskohtaisesti. [11.]

Kuva 13. Esimerkki QlikView-sovelluksella tuotetusta näkymästä [12.]

5.4.4 Palvelutietämyksen hallintajärjestelmä (service knowledge management system, SKMS)

Palvelutietämyksen hallintajärjestelmä kattaa tietokannat ja työkalut, jotka ovat IT-palvelujen tuottajalle elintärkeitä IT-palvelun koko elinkaaren hallitsemiseen. Palvelutietämyksen hallintajärjestelmään on tallennettu muun muassa jokaisen asiakkaan integraatiototeutuksien spesifikaatiot ja osoitteet, integraatiototeutuksissa kohdatut virheet ja niiden ratkaisukuvaukset sekä yleiset periaatteet, miten hallintajärjestelmää ylläpidetään hallitusti.

Esimerkiksi alustariippumaton Atlassian Confluence sekä Atlassian JIRA on usein implementoitu palvelutietämyksen hallintajärjestelmään hoitamaan organisaatiowikiohjelmiston ja tehtävienhallintaohjelmiston virkaa. Niiden ja muiden lisäohjelmistojen avulla on mahdollista ajan kuluessa koota täydelliset toimintalokit ongelmanratkaisusta ja ongelmanratkaisuun liittyneistä hidasteista. Ongelmatilanteen analysointi ei välttämättä aina onnistu täydellisesti vain yhden yksittäisen ilmentymän ansiosta, vaan tilanteen realisoitumiseen on voinut vaikuttaa ilmeisimmän syyn lisäksi jokin taustaprosessi. Täydelliset toimintalokit tarjoavat mahdollisuuden palata aiemman virhetilanteen tulkintaan ja suorittaa sen analyysi uudelleen uuden tiedon avulla, jolloin prosessia voidaan edelleen kehittää.

6 Suunnittelu

Insinöörityössämme suunnittelemme prosessin lokiseurannan QlikView-näytölle, eli suunnittelemme kokonaisen käytännön, mikä jalkautetaan käytettäväksi jatkuvien palveluiden teknisen hallinnan tiimissä. Hyödynnämme työssämme Logstash-sovelluksen lukemia IBM WebSphere MQ -viestijonojen tapahtumalokeja, jotka prosessoidaan ja visualisoidaan lopuksi QlikView-näkymälle. MQ-viestijonojen läpi kulkema data kirjautuu lokiin, josta Logstash-sovellus jäsentää halutun virhe- ja viestidatan työkalun tukemilla ulkoisilla Ruby-skripteillä. Tämän jälkeen data lähetetään MQ-jonoon toisen Logstash-clientin luettavaksi ja edelleen lähetettäväksi QlikViewlle. QlikView taasen prosessoi datan näkymälle. Käytännössä prosessi on vastuussa monen samanaikaisen viestijonon kirjoittaman loki- ja virhedatan seulomisesta, datan analysoimisesta, mahdollisen käyttäytymisen arvioimisesta ja tämän kaiken informatiivisesta visualisoimisesta loppukäyttäjälle.

6.1 Datan keräysprosessin suunnittelu

Prosessin suunnittelussa tuli ottaa huomioon lukuisia eri muuttujia työkalujen yhteensopivuudessa aina arvioitaviin ja suodatettaviin virhetilanteisiin. Pohdimme, tulisiko valmiiksi annettuun malliin lisätä välivaiheita vai vaikeuttaisiko se mahdollisesti prosessia. Prosessiin kuuluvia eri työvaiheita ja työkaluja oli useita, joten se sisälsi myös monta testausta vaativaa osa-aluetta ja mahdollisia sokeita pisteitä. Päätimme sisällyttää siihen vain tarpeellisimmat työkalut yhdessä palvelutietämyksen hallintajärjestelmän kanssa, mitä hyödynsimme prosessissa esiintyvien jatkuvuuskysymysten ratkaisemiseen: Miten toimia virhetilanteessa, jota ei ole tullut aikaisemmin vastaan? Miten varmistaa prosessin toiminta tulevaisuudessa tiimin toimintatapojen tai kokoonpanon muuttuessa?

Alkutilanteessa prosessin työkalujen kokoonpano annettiin meille jatkosuunnittelua varten. Integraatiototeutukset oli tehty käyttäen MQ-jonoja ja niitä lukeva ja suodattava Logstash-client oli asennettu. Logstash-clientin luomien suodatettujen lokikirjausten jonokäsittelyyn tarkoitettu toinen MQ-jono oli myös toteutettu, kuten myös tätä jonoa lukeva Logstash-client, joka toimi yhdessä QlikView-näytön kanssa datan visualisoinnin takia. Saimme myös suodatukseen osallistuvan Ruby-skriptin lähdekoodin läpikäytäväksi ja konfiguroitavaksi, jotta voisimme valita vapaasti suodatettavat rivit. QlikView-näyttöjä oli monitorointitarkoituksessa implementoitu usealle eri jonolle, mutta yksikään

niistä ei ollut tarkoitettu virhetilanteen seurantaan. Jokaisen yksittäisen työkalun toiminta tuli tuntea, jotta niiden välisestä keskustelusta ja oikeasta toiminnasta voitiin varmistua. Työkalut valittiin yhteensopivuus- sekä liiketoimintanäkökulmaa silmällä pitäen, mutta automaattisesti tai lähes automaattisesti toimivien osien lisäksi koko prosessi sisälsi myös inhimillisiä tekijöitä, kuten esimerkiksi sanoman lähettäjän ja toteutuksen valvojan. Nämä seikat, kuten myös prosessin dokumentointi, palvelutietämyksen hallintajärjestelmän ylläpitäminen ja jatkuva oppiminen, tuli ottaa huomioon suunnittelussa. Täten tuli myös osata avata jatkuvien palveluiden teknisen hallinnan, eli ongelmanratkaisu- ja valvontatiimin päivittäinen toiminta, joten aloitimme suunnitteluvaiheen perehtymällä ITIL:n mukaiseen termistöön ja kurssimateriaaliin asiasta. Ensiksi tuli siis opetella kävelemään, ennen kuin kykenimme juoksemaan.

Saimme perehdytyksen ympäristöön ja yllämainitun kurssimateriaalin prosessin suunnittelun avustukseen osallistuvan analytiikkaosaston tiimiltä. Tutkimme, minkälaisia herätteitä ja minkälaisiin virhetilanteisiin Active MQ -ohjelmiston läpi kulkevat sanomat voivat joutua. Opettelimme luomaan omia näkymiä QlikView-ohjelmistolle, jotka olivat prosessin näyttävin elementti ja jota ilman prosessoitu data olisi suodatettu ja generoitu turhaan. Näytölle visualisoitaisiin jonojen virhetilanteet, joita teknisen hallinnan työntekijä tulkitsee ja valitsee oikean menettelytavan palvelutietämyksen hallintajärjestelmän avustamana.

6.2 Datan keräys

Datan keräys ja toteutustapa määritteli kehitettävän prosessin toiminnallisuudet ja datan keräykseen käytettävällä työkalulla päätimme, mitä halusimme seurata ja kuinka laajalti ottaisimme huomioon virhetilanteita. Valvontakohteena olevan MQ-ohjelmiston data päätettiin kerätä jonomanagerien lokitiedostoista. Lokitiedostojen avulla datan keräykselle mahdollistettiin mahdollisimman suuri ja läpinäkyvä aineisto.

```
17/11/2004  10:32:29 - Process(2132.1) User(USER_1) Program(runmqchi.exe)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr (A.B.C)
AMQ9542: Queue manager is ending.

EXPLANATION:
The program will end because the queue manager is quiescing.
ACTION:
None.
----- amqrimna.c : 931 --------------------------------------------------------
```

Kuva 14. Esimerkki MQ-jonomanagerin lokitiedoston yhdestä lokirivistä [13.]

Virhedatan keräämiseen vaaditaan jonomanagerilta, että sen läpi kulkee liikennettä: mikäli liikennettä ei ole, ei ole myöskään valideja liikenteen virhetilanteita, joita tulkita. Sen sijaan liikenteen puuttuminen voidaan nähdä virhetilanteena. Jonomanagerille yleensä määritetään vähintään kerran päivässä suoritettava läpi kulkeneiden sanomien laskuritarkistus, josta voidaan saada selville lähettävän pään tiedonsiirto-ongelma esimerkiksi suojatun yhteyden sertifikaatin vanhetessa. Jonojen määrä jonomanagerilla on skaalautuva, joten laskurin aiheuttamia ”ei liikennettä”-herätteitä käytetään myös ohjelman sisäisesti (kuva 14).

Kuvataan kerättävän virhedatan muodostumista: asiakkuudella seurattavia aktiivisia yksittäisen jonomanagerin jonoja on yhteensä m kappaletta ja jokaisella managerilla on oma virhelokinsa, jonne kaikki jonojen virheet kirjautuvat. Yksittäisen aktiivisen jonon läpi kulkee keskimäärin n sanomaa ajanjaksolla Δt, jonka aikana sanomia managerin läpi kulkee $\frac{mn}{\Delta t}$. Virheitä aktiivisista jonomanagereista muodostuu tällöin ajanjaksolla keskimäärin,

$$v = \frac{cmn_t p}{\Delta t} ,$$

missä c on jonomanagerien määrä, m on jonomanagerin jonojen määrä, n on yksittäisen jonon läpi kulkeneet viestit ja p on järjestelmän virheen muodostaneiden sanomien ja kaikkien aktiivisen järjestelmän läpi kulkeneiden sanomien osamäärä, eli ns. virhealttius.

Jonomanagerien jonojen määrän ollessa skaalautuva ja jonomanagerien määrän ollessa sanomaliikenteen kokoon mitoitettu, jokainen tarvittava jonomanageri tarkoittaa huomattavaa sanomaliikenteen kasvua, mikä puolestaan lisää sanomavirheitä lokidatassa.

Lokidatan keräykseen käytettävää ohjelmistoa valitessa vertailimme eri ohjelmistoja muokattavuuden sekä liiketoimintanäkökulman perusteella. Lokidatan keräykseen sovellettiin avoimen lähdekoodin Logstash-ohjelmistoa. Logstash on monipuolinen ohjelmisto, jolla voidaan rakentaa omaan tarkoitukseen muokattuja skriptejä datan keräykseen. Avoimen lähdekoodin ohjelmistona Logstash on myös hyvin muokattavissa sekä kustannustehokas, joten nämä olivat osasyynä ohjelmiston valintaan yhdessä yhteensopivuuden QlikView-näytön kanssa.

Logstash-ohjelmiston clientit asennettiin jokaisen jonomanagerin palvelimelle. Palvelimille rakennetut Ruby-skriptit suodattivat lokeista haluttua dataa. Skriptillä kerätty lokidata lähettiin sovellukselle rakennetun sanomanjonon kautta eteenpäin tiedostopalvelimelle, jossa seuraava Logstash-client luki datan jonosta ja kirjoitti datan QlikView-ohjelmistolle sisäänluettavaan muotoon.

6.3 Näkymän suunnittelu

QlikView-näkymän perusteellinen suunnittelu oli näkymän loppukäyttäjää ajatellen tärkeää, sillä dataa oli tuleva olemaan paljon. Lisäksi se tulisi monitoroida mahdollisimman harvalle näkymälle niin, että teknisen hallintatiimin jäsen pystyisi nopealla vilkaisulla sisäistämään tilanteen ja päättelemään ongelma- tai virhetilanteen laadun sekä sitä seuraavan toimenpiteen. Näkymä on prosessissa keskeinen työkalu, sillä kaikki tiimin tekemät toimenpiteet perustuvat näkymän antamaan informaatioon.

Qlikview-näkymän suunnittelussa kannattaa käyttää hyväksi todettuja käyttöliittymäsuunnittelun periaatteita. Suunnittelun hyvänä lähtökohtana on tuntea kohderyhmä, jolle käyttöliittymä tulee käyttöön, sillä on hyvä selvittää, minkälaisia käyttöliittymiä käyttäjällä on käytössään parhaillaan ja ottaa ne huomioon näkymää suunnitellessa. Käyttöliittymästä on tärkeää tehdä sellainen, jota käyttäjä on aikaisemmin tottunut käyttämään, jotta käyttöliittymään perehtyminen sujuisi vaivatta. Näin käyttäjä tuntee uuden käyttöliittymän kotoisaksi ja oppii käyttämään sitä nopeammin. Uutta käyttöliittymää suunnitellessa on otettava huomioon myös johdonmukaisuus: käytön tulisi olla ennakoitavaa. Samankaltaisten valitsimien tulee sisältää yhtäläinen toiminnallisuus ja eri tavalla toimivat valitsimet tulisi erottaa selkeästi.

Ennen kaikkea käyttöliittymän tulisi olla tarpeeksi yksinkertainen ja helppokäyttöinen. Tarvittavat ja hillityt visuaaliset kontrastit lisäävät selkeyttä ja ohjaavat käyttäjää tärkeisiin toiminnallisuuksiin. Käyttäjälle on annettava palautetta tehdyistä toiminnoista, jotta käyttäjä tietää toimineensa oikein tai väärin. Palautteen lisäksi on tärkeää antaa käyttäjälle anteeksi mahdolliset virhetilanteet: On annettava mahdollisuus palata edelliseen tilanteeseen.

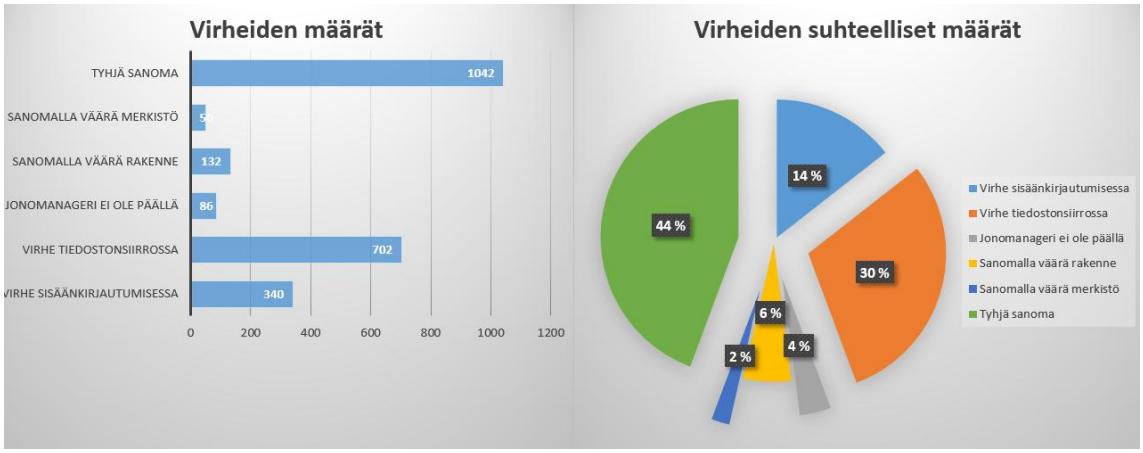
Kehitimme valvontanäkymää yhdessä teknisen hallinnan tiimin työntekijän kanssa ja kuulimme valistuneen mielipiteen datan oikeanlaisesta esittämisestä ja miten näkymästä saisi mahdollisimman tehokkaan ja informatiivisen juuri heille. Tietyt tavat datan visualisointiin ovat tietorikkaampia kuin toiset: pylväsdiagrammiin katoaa usein dataa, kun taas viivadiagrammiin saadaan merkittyä aikamerkinnät tarkemmin, sillä merkintöjä ei ryhmitellä tietyn ajanjakson mukaan. Näkymän suunnittelussa on myös tärkeää ottaa huomioon, mihin tarkoitukseen näkymää suunnitellaan. Johtoryhmille on tärkeämpää datan oikeanlainen asettelu ja yhteyksien epäkohtien yhteenlaskettu lukumäärä sekä palvelutasosopimuksen toteutumisaste, ja tekniselle hallintatiimille tärkeämpää on yksittäisen virheen päätepisteet, virheen aiheuttanut solmukohta ja mahdollinen ratkaisu ongelmaan.

Eri prosesseja ja projekteja arvioidaan KPI (key performance indicator) -tehokkuuden mittausasteikoilla, joita voidaan määritellä ja soveltaa myös datan esittämisen arviointiin. Yksiselitteinen ja kattava näkymä toimii vain yhdellä määrätyllä tavalla oikein. Väärällä tai liian monimutkaisella tavalla esitetty tilasto ei ole ainoastaan tietoarvoltaan vähäinen, vaan tilastollisesta datasta voidaan muodostaa suoranaisia valheita. Kolmiulotteiset kuvaajat voivat vääristää tilannekuvaa ja väärin suodatetusta datasta on mahdollista määrittää epäolennaisia korrelaatioita, mikäli jotain tärkeää jää ilmoittamatta. Esimerkiksi epäonnistuneita sisäänkirjautumistapahtumia ei ole tapahtunut yhtäkään kappaletta, mikäli sisäänkirjautumistapahtumia ei ole yhtäkään kappaletta.

Paljon tietoa sisältävien kuvaajien kääntöpuolena on usein niiden vaatiman tilan koko. Mikäli näkymään tulisi asettaa monia eri tietoalkioita tai samojen tietoalkioiden eri visualisointeja on myös tilan hyötykäytöllä merkitystä, ettei näkymien lukumäärä kasva tarpeettoman suureksi. Näkymien rakenne voisi olla myös kuvaileva ja kerrostunut, jotta sen avulla voidaan ohjailla työntekijän työprosessia. Eri kuvaajien yhdistely esimerkiksi

niin, että yleisnäkymässä käytetään piirakkadiagrammia, jonka jälkeen alemmilla yksityiskohtaisemmilla tasoilla käytetään tarkempaa tietoa sisältävää ja ryhmittelemätöntä tietoa.

Näkymien rakenne ja niiden asettelu on oltava suhteessa näytettävään dataan. Määrältään tiettyä tyyppiä edustava data ei sovi esitettäväksi järkevästi kaikilla diagrammeilla ja asteikoilla. Pituushyppääjien suoritteita ei ole järkevä visualisoida piirrakkadiagrammille eikä ryhmitellä pylväsdiagrammille, mikäli tahdotaan nähdä erot yksittäisissä suoritteissa.



Kuva 15. Esimerkki tietoarvoltaan käytettävissä olevasta (vas.) sekä epäolennaisesta kuvaajasta (oik.), jotka on luotu samasta lähtöaineistosta. Vasemmassa kuvaajassa käyvät ilmi virheiden laatujen määrä ja kokoluokka sekä niiden välinen suhde. Oikeanpuolisesta kuvaajasta ilmenee, että virheitä on tapahtunut, mutta kokoluokasta ei välity mitään tietoa.

Esitettävä data tulee suodattaa hyvin ja purkaa sen eri rakenneosat erilleen, jotta niitä voidaan ryhmitellä uudelleen. Uudelleenryhmittelyssä on tärkeää löytää oikeat korrelaatiot ja juurisyyt, jotta lopullisia yksiselitteisiä arvoja on mahdollista muodostaa. Esimerkiksi sisäänkirjautumisen epäonnistuminen voi johtua toistuvasti eri muuttujista: yhteys tunnusten validointipalvelimeen voi olla katki tai käyttäjä voi syöttää väärän käyttäjätunnus-salasana-parin ja sisäänkirjautuminen epäonnistuu tästä syystä, vaikka palvelun käyttäjän analyysi tilanteesta olisikin molemmissa tapauksissa ”yhteys ei toimi”.

6.4 Prosessin suunnittelu

Prosessin suunnittelussa lähdimme ensin liikkeelle oikean virhedatan suodattamisesta, ongelmatilanteen havaitsemisesta sekä häiriöiden luokittelun määrittelystä. Opinnäytetyössä kohteena ja ratkaistavana ongelmana oli suunnitella MQ-sovellukseen suurella kapasiteetilla ilmaantuvien virhetilanteiden tunnistamiseen ja ennaltaehkäisyyn prosessi ja samalla suunnitella prosessin optimaalinen käyttäminen. Suunnitteluvaiheessa pyrimme kartoittamaan ja ottamaan huomioon prosessin mahdolliset kompastuskivet ja sen tulevaisuudessa kohtaamat haasteet tarkkaan. Kehitettävä prosessi tulisi käyttöön jatkuvien palvelujen toiselle tasolle (Tekninen hallinta), joten sen käytöllä ja huolimattomalla suunnittelulla tulisi olemaan myös liiketoiminnallista vaikutusta.

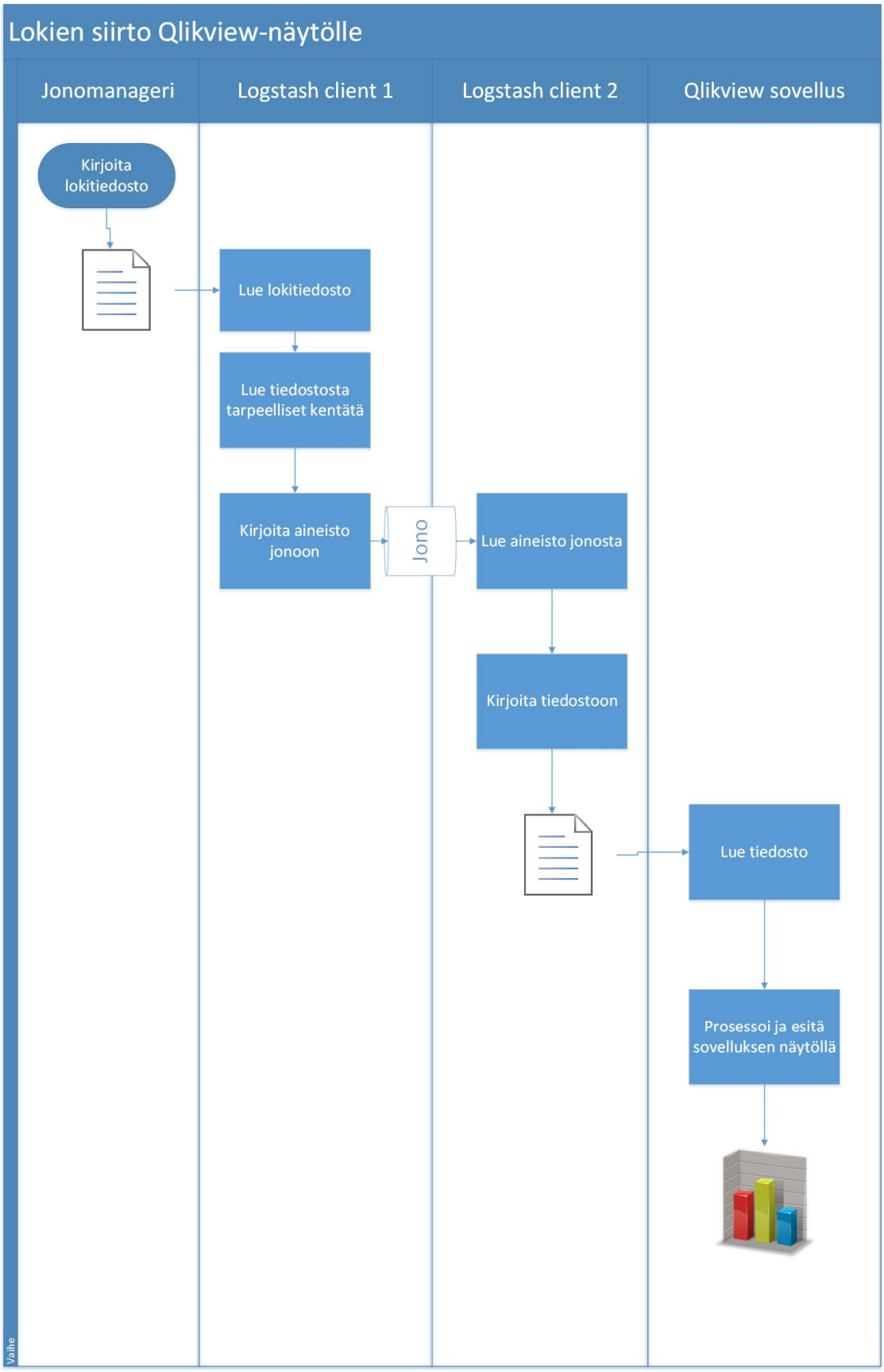
Suuri haaste kehitettävälle prosessille oli tiedon puute aikaisempien virhetilanteiden syistä ja seurauksista, joita lähdimme kartoittamaan jonojen kirjoittamien lokien ja nykyisen käyttäjäkunnan tietojen pohjalta. Kehitysvaiheessa tiesimme, että prosessille oli annettava lähtökohdat nopeaan oppimiseen, mutta vasta ensimmäisten suurten virhetilanteiden jälkeen prosessi tulisi saavuttamaan sille asetetut tavoitteet. Prosessi on tarkoitettu jatkuvasti valvottavaksi ja kehitettäväksi, jotta sen käyttäminen parantaa valvonnan laatua ja valvonnalle asetetut tavoitteet saavutetaan yhä pienemmillä resursseilla.

Prosessia suunnitellessa pyrimme saamaan QlikView-näkymän yksinkertaiseksi ja informatiivisuudeltaan mahdollisimman kattavaksi. Tämän jälkeen prosessia pystyttiin lähteä kehittämään oikeaan suuntaan. Prosessin kehityksessä pyrimme luomaan ja lopulta jalkauttamaan päivittäin toteutuvat toimenpiteet, joiden mukaan rakennettua QlikView-näkymää seurattaisiin ja mihin asioihin tulisi kiinnittää huomiota. Tämän lisäksi huomasimme, että prosessin tueksi olisi hyvä ottaa käyttöön virheenhallintarekisteri tulevia virhetilanteita ja niiden ennaltaehkäisyä varten.

Päätimme liittää prosessin mukaan palvelutietämyksen hallintajärjestelmän alaiseen toimintaan. Palvelutietämyksen hallintajärjestelmään kuuluvien työkalujen avulla virhetilanteiden ratkaisukuvausten ylläpitäminen ja hallinta tehdään helpommaksi. Lisäksi tiedot edellisistä häiriötilanteen ilmentymistä ovat talletettuina uuden kohdatun tilanteen selvittämistä varten.

7 Prosessin kehitys ja jalkautus

Insinöörityössä suunnitellaan prosessi ja prosessin käyttämä QlikView-näyttö, jono-managerien kirjoittamasta lokidatasta analysoidun tiedon hyötykäyttöön. Integraatiototeutuksen lävitse kulkevasta dataliikenteestä kirjautuu lokitietoihin niin kaiken tyyppiset herätteet, häiriöt kuin hälytykset. Lokidataa lukemalla erillisen työkalun avulla saadaan luettua seuraavalle työkalulle tarvittava data tiedon analysoinnin ja prosessoinnin jälkeen lopulta tilannekuvan visualisointiin niin virheiden kuin normaalin toiminnan osalta (kuva 16).



Kuva 16. Lokien siirto QlikView-näytölle prosessikaaviona

Prosessin alkutilanne määritti jo useimmat käytettävät työkalut, sillä ei olisi tarkoituksenmukaista tai tarpeellista lähteä muuttamaan jo yhtiön sisällä vallitsevien käytäntöjen kanssa yhteensopivaa kokoonpanoa. Integraatiototeutus IBM MQ -ohjelmiston avulla oli jo valmiina, joten jonomanagerien lokidatan kirjoitus oli myös toteutettuna. Tämän jälkeen kirjoitettu loki luettiin Logstash-ohjelmistolla ja siitä suodatettiin haluttu data Elasticsearch-hakumoottorin avulla. Elasticsearch-hakumoottorin avulla MQ-sovelluksen tuottaman lokidatan (kuva 14) parsiminen onnistui vaivattomasti. Vaivattomuuden lisäksi Elasticsearch-hakumoottori tukee huomattavan määrän lisäosia, joiden avulla haettu data saatiin muokattua suoraan haluttuun muotoon vastaanottavassa päässä.

Päätös käyttää QlikView-sovellusta datan visualisointiin oli selvä jo alkujaan, sillä kyseisellä asiakkuudella oli käytössään useita muitakin QlikView-sovellusta hyödyntäviä prosesseja. Täten sovellus oli tuttu suurimmalle osalle kohderyhmän työntekijöistä. Sovelluksen ulkoasun muuttaminen tekee usein monitorointityökalusta aivan eri ohjelman loppukäyttäjän mielestä, joten näimme meillä kuitenkin olevan valtaa myös toteutuksen visuaaliseen lopputulokseen. QlikView-sovelluksen hyödyntäminen opinnäytetyön projektissa oli perusteltua myös sen skaalautuvuuden ja etenkin muokattavuuden takia.

Prosessin kehitys toteutettiin neljässä vaiheessa:

1. Ensimmäisessä vaiheessa lähdettiin selvittämään prosessin tarpeita eri tiimeille ja käytettävissä tai jo käytössä olevia työkaluja prosessin tueksi. Useat prosessin kehityksen tueksi valitut työkalut olivat jo aikaisemmin käytössä jollain tiimillä ja täten niiden käyttö oli osittain tuttua.

2. Suunnitelmaa lähdettiin hyödyntämään prosessin varsinaisessa kehitysvaiheessa. Analytiikka- ja visualisointityökaluksi valitun QlikView-näytön pohjaa lähdettiin kehittämään ensimmäiseksi. QlikView-näyttö pyrittiin saamaan pilottiin jo varhaisessa vaiheessa, jotta kehityksen tueksi saataisiin myös itse valvontatyötä tekevien henkilöiden palautteet.

3. Pilotointi toteutettiin pienellä loppukäyttäjiä vastaavalla ryhmällä. Pilottiryhmän käyttöön annettiin kehitysvaiheessa oleva näkymä ja sen käytössä ohjeistettiin lyhyesti ennen varsinaista pilotointia. Pilotoinnissa tuli vastaan muutamia kehitysideoita ja palautteita, joita otettiin mukaan lopullisen näytön kehityksessä.

4. Viimeisessä vaiheessa prosessin kehitystä QlikView-näytön hyödyntämiselle valvontatyössä kehitettiin ohjeet, jonka mukaan valvontatyötä tekevät henkilöt osaavat reagoida oikeisiin tilanteisiin ja eskaloida tilanteen vaatiessa.

Toteutuksen tuloksena saatiin prosessi, joka ohjaa työntekijöitä integraatioiden valvonnassa ja tuottaa helposti ja nopeasti ymmärrettävään muotoon pakatun tiedon kunkin

hetken tilanteesta integraatioympäristössä. Opinnäytetyössä kehitetty prosessi on käyttöönotettaessa puutteellinen. Prosessi on luotu hyödyntämään ennakkotilanteita ja prosessin hyödyt tulevat esille kun ennakkotapauksia alkaa kertymään. Prosessin hyödyn parantamiseksi virhetilanteiden kerääminen palvelutietämyksen hallintajärjestelmään on olennaisessa osassa prosessin jatkuvaa kehittymistä.

7.1 Lähtökohdat

Prosessin kehityksen lähtötilanteessa oli olemassa tiimit, joille prosessi kehitettäisiin. Prosessin kohderyhmänä on kaksi tiimiä. Tiimit ovat olleet kasassa jo pitkään ja niiden välillä on ollut pidempään yhteistyötä erilaisten aikaisemmin jalkautettujen prosessien ympärillä. Täten lähtökohdat prosessin kehitykselle olivat hyvät, koska molemmilla tiimeillä oli jo ennestään kokemusta erilaisten valvontaprosessien toimintatavoista. Prosessin suunnitteluvaiheessa päätetyistä käyttöön otettavista työkaluista: Logstash, QlikView ja IBM MQ, tiimeillä oli kattava kokemus. Valituista työkaluista kuitenkin Logstash- ja QlikView-sovelluksen hyödyntämistä valvontatyössä haluttiin korostaa entisestään kehitettävän prosessin jalkautuksen yhteydessä.

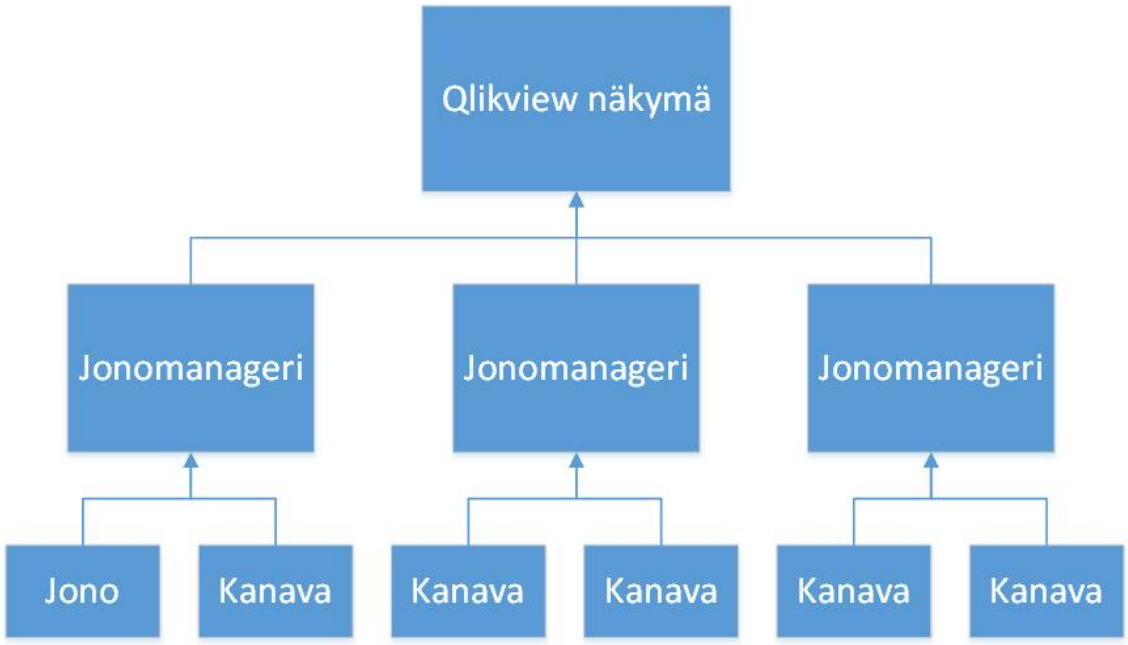
Koska prosessiin valittuja työkaluja oli hyödynnetty aikaisemmin niin integraatioissa kuin eri valvontaprosesseissa, ei prosessin kehityksessä tarvinnut asentaa kyseisiä sovelluksia vaan sen sijaan olemassa olevia ympäristöjä laajennettiin. Logstash clientin lokien lukua kehitettiin vastaamaan paremmin kyseessä olevan prosessin kehityksen tavoitetilaa. Tämä tarkoitti käytännössä sitä, että Logstash sovelluksen hyödyntämää Elasticsearch-parsintaa karsittiin hieman ja lokitiedostosta otettiin jatkokäsittelyyn vain tarvittavat kentät. Tässä yhteydessä käytetty jonoyhteys lokien siirtoon, logstash clientien välillä oli rakennettu jo aikaisemmin, eikä tähän tarvittu muutoksia prosessin kehitysvaiheessa.

QlikView-sovellus oli olemassa asiakaan ympäristössä ja siihen oli rakennettu useampia näyttöjä muiden prosessien tarkoituksiin. Näitä valmiita näyttöjä ei pystytty hyödyntämään eri datan takia vaan kehitettävälle prosessille lähdettiin rakentamaan uutta näyttöä.

7.2 QlikView-näkymän kehitys

QlikView-näkymän kehitys on oleellinen osa kehitettävää prosessia. Prosessin mukaan toimivat työntekijät hyödyntävät kehitettyä näkymää virheiden seurannassa ja näkymän on tarkoitus visualisoida tarvittavat tiedot valvonnan nopeuttamiseksi ja helpottamiseksi. Helpon ja nopean muokattavuuden takia tuleva QlikView-näyttö kehitettiin hyödyntäen QlikSense-sovellusta. QlikSense on saman yrityksen kehittämä QlikView-sovellusta vastaava pilvipohjainen ohjelmisto näkymien rakentamiseen.

QlikView-näkymää suunnitellessamme lähdimme miettimään parhainta mahdollista tapaa virhelokista kerättyjen tietojen esittämiselle. Miten valvontatyötä tekevä henkilö saisi parhaiten irti tarpeellisen tiedon näkymästä ja millä visuaalisilla keinoilla näytöstä saataisiin tarpeeksi selkeä?

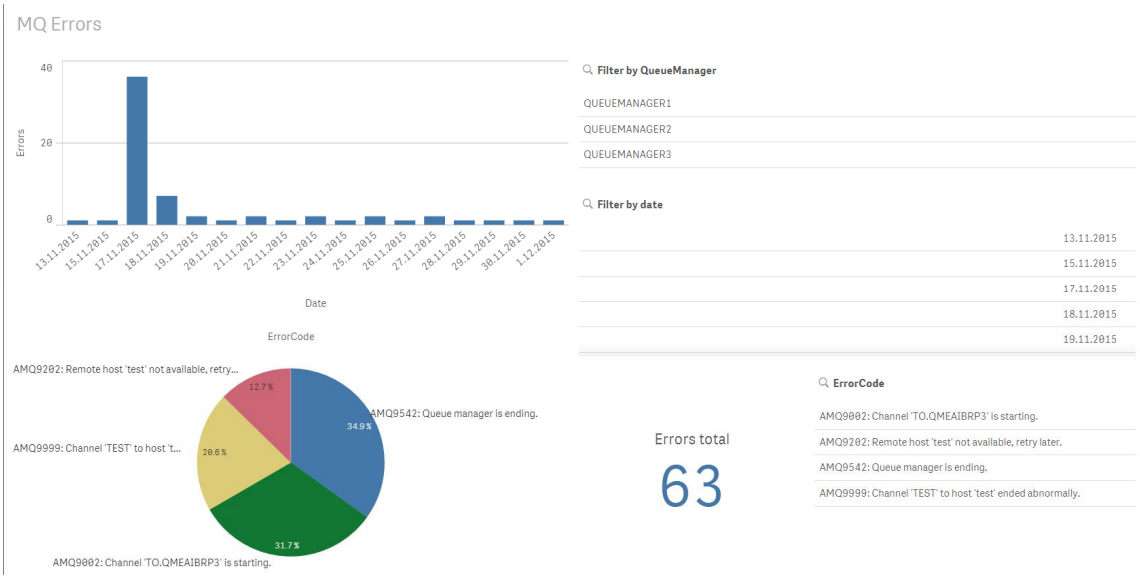


Kuva 17. Virhedatan keräys QlikView-näkymälle

Jonomanagereiden alla on kanavia ja jonoja, joihin virheet pääasiassa kohdistuvat. Suunnittelun alkuvaiheessa mietittiin, kuinka tarkasti virheiden taso tulisi näyttää. Päädyimme siihen, että jonojen ja kanavien virheitä ei erotella, vaan käyttäjälle näytetään

virheet jonomanagerien tasolta. Jonomanagerien alta löytyy erikseen kyseisen jonomanagerin jonojen ja kanavien virheet (kuva 17). Mikäli asiakas haluaa nähdä tarkemmin tietyn jonon virheet, voi hän tarkistella niitä tarkemmin lokitiedostoista kyseisiltä palvelimilta. Näin käyttäjälle saadaan yksilöityä virheen aiheuttanut jonomanageri ja nopeutetaan virheen selvittämistä näyttämällä käyttäjälle vähemmän dataa, josta päätellään ongelman laatu. Näkymän seuraaminen tulisi olla selkeää ja helposti omaksuttavaa jokaiselle käyttäjälle, joten näkymien selkeys ja yksinkertaisuus korostuivat näytön kehityksessä. Virhedata päädyttiin esittämään monen eri näkymän avulla. Eri näkymien avulla kuvaajista saadaan selkeitä ja yksiselitteisiä. Havaitessaan poikkeustilanteen käyttäjä pystyy siirtymään seuraavalle näytölle, jossa valittu data näytetään tarkemmalla tasolla.

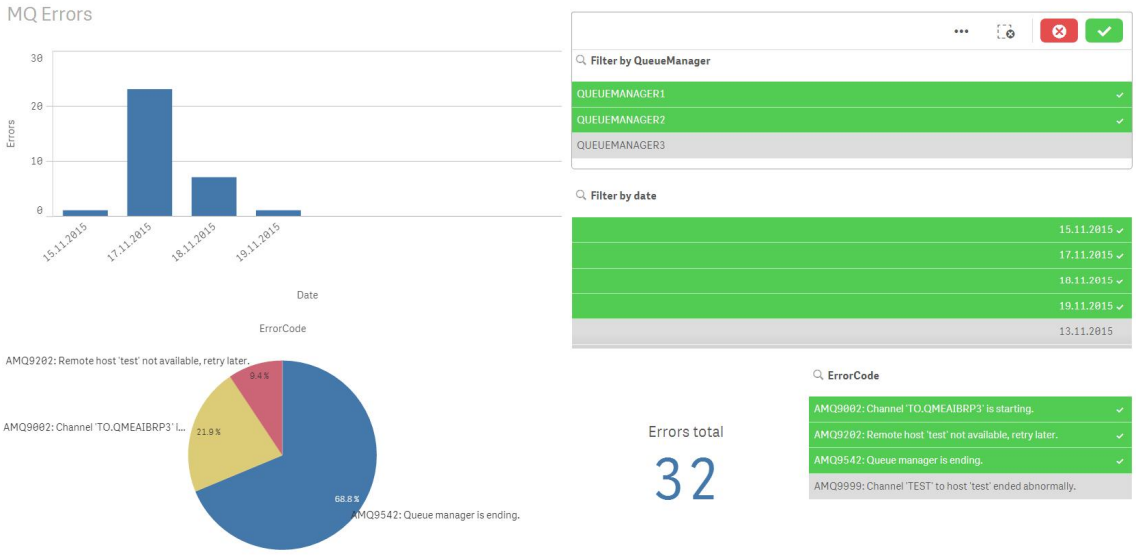
Ensimmäinen käyttäjälle avautuva näkymä on yleisnäkymä, jossa käyttäjälle pyritään antamaan mahdollisimman kattavasti informaatiota (kuva 17). Näytöllä käyttäjälle näytetään virheiden kokonaismäärä päivämäärien mukaan, virheiden jakautuminen piirakkakaavion avulla, virheiden lukumäärä kokonaisuudessaan ja kaikki eri virhekoodit.



Kuva 18. QlikView-sovelluksen alkunäkymä

Käyttäjällä on mahdollisuus rajata näkymässä näkyviä tuloksia jononmanagerien mukaan, valitsemalla jonomanageri suodatusvalikosta. Lisäksi näkymän tuloksia on mahdollista myös rajata haluamansa päivämäärät omasta suodatusvalikosta sivun oikeasta

laidasta. Näkymän kautta käyttäjä pystyy erottelemaan mahdolliset päivämäärät ja solmukohdat, jolloin poikkeustapauksia on tapahtunut eniten. Tällöin käyttäjälle näytetään pelkästään kyseisen jonomanagerin virheet tietyille päiville (kuva 14).

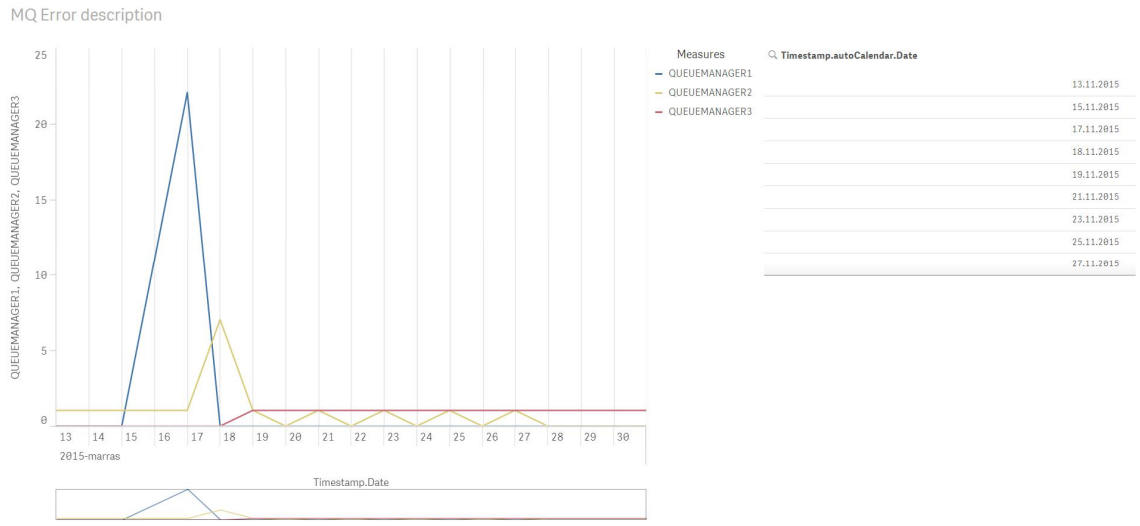


Kuva 19. Esimerkki rajauksista alkunäkymässä

Aloitusnäkymä rajattunakin on tiedonannoltaan ylimalkainen. Tarkempaa tietoa saadakseen on käyttäjän siirryttävä seuraavaan näkymään, johon hän pääsee painamalla virheiden lukumäärään upotettua linkkiä (kuva 19).

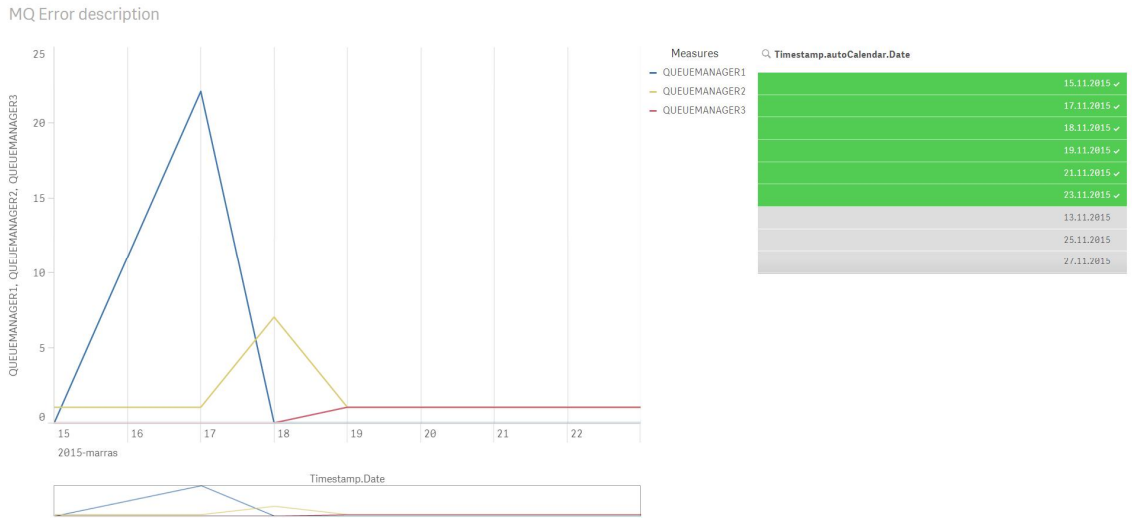
Kuva 20. Painamalla virheiden lukumäärää päästään seuraavalle QlikView-näkymälle.

Toisessa näkymässä käyttäjä saa selville tarkemmin valitsemansa jonomanagerin virheiden jakautumisen.



Kuva 21. Näkymä 2 virheiden lukumäärä viivadiagrammina.

Näkymällä data näytetään virheiden lukumäärä kullekin ajankohdalle viivadiagrammina (kuva 21). Diagrammissa jokaiselle jonomanagerille on oma viiva. Tämän avulla käyttäjä pystyy näkemään tietyn jonomanagerin virheiden lukumäärän tietyllä ajanjaksolla ja virheiden ajankohdan. Näkymässä käyttäjällä on mahdollisuus rajata diagrammia päivämäärien mukaan, jolloin käyttäjä näkee virheiden tarkemman ajankohdan (kuva 22). Tämän näkymän tarkoituksena on antaa käyttäjälle tarkempaa tietoa mahdollisen virhetilanteen paikantamiseksi ja selvittämiseksi.



Kuva 22. 2. näkymän tulosta rajattu päivämäärän mukaan.

Lopputuloksena saadaan näyttö, joka toimii kahdessa eri tasossa. Ensimmäisessä tasossa annetaan käyttäjälle yleiskuva virhetilanteiden valvonnasta ja tarvittaessa tilannetta voidaan tarkastella tarkemmin toisella näytöllä.

7.3 Prosessin kehitys

Ilman prosessia edellä kehitetyn näkymän seurannasta ei saataisi kaikkea hyötyä irti. Etenkin tässä tapauksessa prosessina toimivat ohjeet, joiden avulla kehitetyn QlikView-näkymän tilanteisiin reagoidaan. Tämän prosessin tavoitteena on tehostaa valvontaa MQ-virhelokien erilaisten tilanteiden seurantaan. Prosessikehityksen tuloksena saaduissa ohjeissa mainitaan, miten ja millä aikaväleillä näkymää seurataan, miten eri tilanteisiin reagoidaan ja mahdollisia raja-arvoja seurannassa huomattujen virhetilanteiden eskaloimiseen. On kuitenkin muistettava, että prosessin alkuvaiheessa raja-arvojen ja virhetilanteiden tunnistaminen on haastavaa ennakkotilanteiden puutteellisuuden vuoksi. Täten prosessissa määritellyt ohjeet kehittyvät jatkuvasti ennen käyttöönottoa ja etenkin käyttöönoton jälkeen, ennakkotapausten lisääntyessä. Prosessin kehittämisen apuna käytettiin jo yrityksen sisällä toiminnassa olevien valvontaprosessien hyväksi todettuja toimintatapoja.

Prosessissa valvotaan QlikView-näkymää virhetilanteiden määrän seuraamiseksi. Prosessi on suunniteltu etenkin tukipalvelujen toisen tason valvontatyökaluksi, josta eskaloidaan epänormaaleja tilanteita kolmannelle tasolle. Tarkoituksena on työvuoron aluksi tarkastaa etenkin edellisen päivän tilanne QlikView-näkymältä. Onko huomattavia muutoksia virheiden määrässä verraten edellisen muutaman päivän määriin? Mikäli merkittäviä muutoksia huomataan yhdellä tai useammalla jonomanagerilla, katsotaan toiselta näkymältä tarkemmin, mille ajankohdalle kyseiset virhetilanteet ajoittuvat. Kun mahdollisen virhetilanteen ajankohta saadaan selville, selvitetään, mistä virhetilanteet johtuvat. Tähän osviittaa antaa ensimmäisen näkymän virhetyypit sekä mahdolliset huolto- ja häiriöilmoitukset. Virhetilanteessa olennaisena osana on myös seurata, onko kyseessä olevan jonomanagerin toiminta normaalia vai havaitaanko käyttöliittymältä esimerkiksi jonoja, joissa on huomattava määrä sanomia jumissa. Mikäli virhetilanteen syytä ei saada paikannettua toisella tukipalvelun tasolla, eskaloidaan tilanne tukipalvelujen kolmannelle tasolle kehitystiimin tietoon.

7.4 QlikView-näkymän arviointi

Arvioinnin tarkoituksena on testata näkymän toimintaa osana valvontaprosessia ja tunnistaa siihen liittyviä mahdollisia kehitystarpeita. Tavoitteena on myös edistää valmiin näkymän ja valvontaprosessin jatkuvaa kehittämistä, käyttöönottoa ja edelleen kehittämistä.

QlikView-näkymän ensimmäisen kehitysvaiheen jälkeen valittiin pieni, lopullista kohderyhmää vastaava ryhmä, jolla näkymää ja sen käyttöä testattiin käytännössä tuotantoa vastaavalla suppealla datalla. Näkymän arviointiin tehty näyttö toteutettiin aikaisemmin mainitulla QlikView-sovellusta vastaavalla QlikSense-sovelluksella. Näkymän pilotoinnin tarkoituksen oli testata kehitetyn prototyypin toimintaa oikealla kohderyhmällä.

Arviointiin valmistauduttiin valitsemalla oikea kohderyhmä ja aineisto, jolla näkymää testattaisiin. Aineisto pyrittiin valmistelemaan niin, että jokainen mahdollinen tapaus alkuvaiheessa tulisi käytyä läpi ja toimivuutta saataisiin arvioitua laajasti. Toisaalta myös se, että pilotoinnin tuloksena saataisiin huomattava määrä palautetta kohderyhmältä näkymän jatkokehitystä varten.

Ennen arviointia kohderyhmä perehdytettiin näkymän käyttöön. Käyttäjille esiteltiin näkymät, niiden komponentit ja kunkin komponentin ja näkymän käyttötarkoitus. Testiryhmälle pilotoinnin tueksi tehtiin lisäksi tukimateriaalia, johon listattiin toiminnallisten asioiden lisäksi asioita, joihin tulisi kiinnittää huomiota. Täten arvioinnista saataisiin irti mahdollisimman kattavaa palautetta ja kehitysideoita.

Itse arviointi toteutettiin antamalla jokaiselle kohderyhmän työntekijälle käyttöön kehitetty QlikView-näkymä. Käyttäjille annettiin muutama tehtävä, johon pitäisi pystyä vastaamaan näkymän avulla. Samalla käyttäjien tulisi arvioida näkymän selkeyttä ja toiminnallisuutta, kirjoittaa palautetta etenkin mahdollisista puutteista ja kehityskohdista. Näitä palautteita hyödynnettäisiin myöhemmässä vaiheessa kehitystä.

Prosessin jatkuvan oppimisen tueksi on ylläpidettävä palvelutietämyksen hallintajärjestelmää (wiki). Järjestelmään päivitetään kaikki uudet virhetilanteet ja ohjeet niiden selvitykseen.

7.5 Prosessin pilotointi

Suunnittelimme prosessin käytettäväksi kahdessa jatkuvien palveluiden teknisen hallinnan tiimissä. Suunnittelemamme käytäntö edusti kuitenkin suurimmilta osin vain keräämäämme tietoa ja omaksuttua näkemystämme valvonnan teoreettisesta toiminnasta. Päätimme ennen varsinaista jalkautusta pilotoida prosessia tiimin varsinaista kokoa pienemmälle testiryhmälle, jotta kuulisimme myös valmiin prosessin tulevien käyttäjien mielipiteitä ja näkemyksiä koskien muun muassa näkymän asettelua, kompastuskohtia ja sitä, miten suunniteltu prosessi esimerkiksi tulisi sulautumaan yhteen päivittäisen työn oheen.

Prosessissa valvotaan integraatiototeutuksen sanomapohjaisen väliohjelmiston läpi kulkeneiden suoritteiden muodostamien, suodatuksen ja analysoinnin kautta edelleen ohjattujen lokikirjausten visualisointia erillisille näkymille, joita tarkkailemalla on mahdollista tehdä johtopäätöksiä integraatiototeutuksen statuksesta ja mahdollisista jatkotoimenpiteistä. Valitsimme sekä asiakaskohtaisesta että yleisten jatkuvien palveluiden tiimistä yhden jäsenen mukaan pilottiin. Pilotointi toteutettiin työtehtävistä eriytettynä.

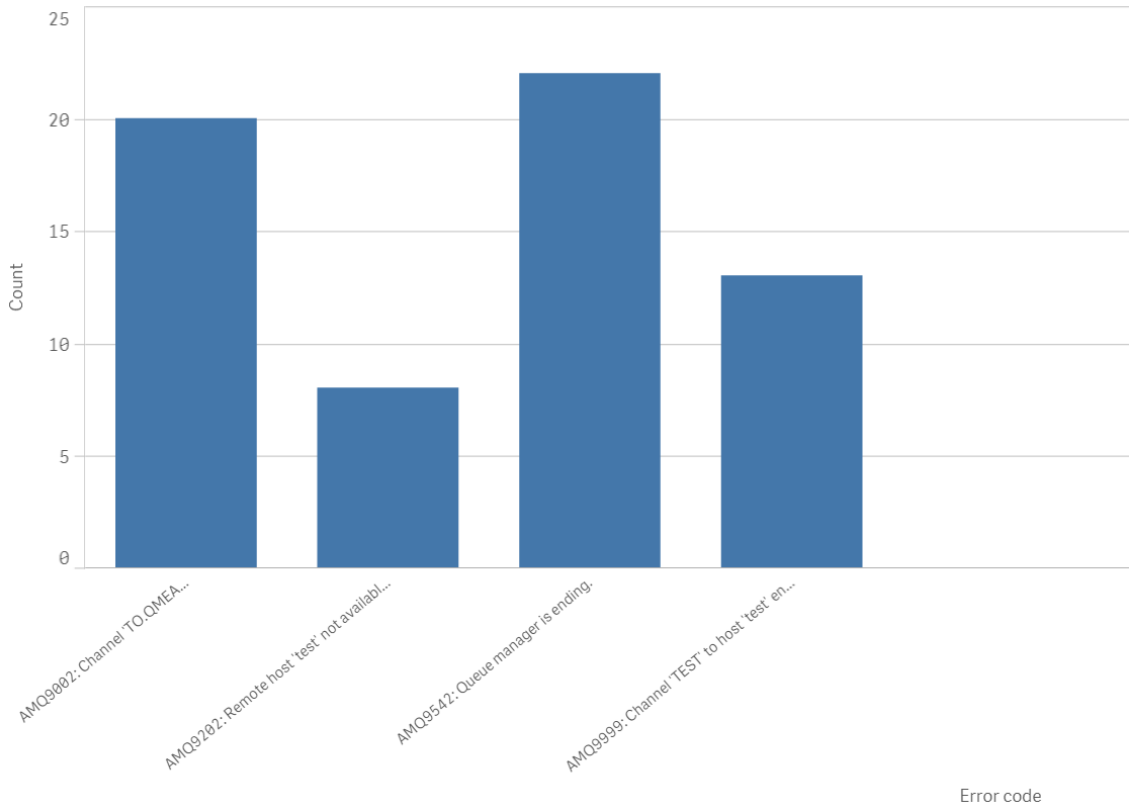
Pilotoinnissa kävimme lävitse prosessin alaisen toteutuksen ja datan keräämiseen käytettävät kriteerit. Toteutuksen selventäminen kävi helposti, sillä molemmissa tiimeissä oli samalla teknologialla toteutettuja muita sanomavirtoja, joten valvontaprosessin assimilointi tapahtui kivuttomasti. Ajatus monitoroidusta näkymästä oli tiimien jäsenten mukaan mieluinen eikä valvontaprosessia nähty liialti muita työtehtäviä kuormittavana. Saimme palautetta kuitenkin näkymien suunnittelusta: yleisen jatkuvan palvelun tiimille olisi mielekkäämpää käyttää näkymiä, mikäli ne muodostaisivat palvelupolun ja näkymät olisi toteutettu kerrostuneesti. Ensimmäisestä näytöstä ei tulisi käydä ilmi kaikkea oleellista, vaan epäinformatiivisesta näkymästä pääsisi valitsemaan seuraavan näytön, joka visualisoisi valitun raportin yksityiskohtaisemmin. Näin näkymien käytöstä tulisi ohjailtua ja estettäisiin ”tietoähkyä”, liiallista informaatiotulvaa yksittäisen näkymän käytöstä. Parannusehdotus oli myös asiakaskohtaisen tiimin edustajan mielestä toimiva, joten päätimme muuttaa toteutusta pilotoinnista saatujen palautteiden perusteella.

7.6 Palautteet ja jatkokehitys

Pilotoinnin tärkeimpänä tarkoituksena perehdytyksen lisäksi oli saada palautetta prosessista ja näkymästä sekä niiden yhteistoiminnasta. Palautteiden avulla prosessiin ja näkymään tehdään muutoksia. Palautteet ovat tärkeitä siinäkin mielessä, että oikeat käyttäjät näkevät asiat eri näkökulmasta kuin itse prosessin kehittäjät. Prosessi kehitetään valvontatyötä tekevien työntekijöiden käyttöön, joten on tärkeää, että näkymä ja prosessi palvelevat kohderyhmän tarpeita.

Arvioinnin ja pilotoinnin avulla saatiin kattavasti palautetta ja kehitysideoita niin näkymän kuin prosessin jatkokehitykseen. Näkymässä ajatuksia herätti etenkin näytettyjen kaavioiden tarkkuus. Käyttäjät kaipasivat näkymää, jolla näytettäisiin vielä tarkemmin virheiden ajankohta sekä kyseisen ajankohdan virhekoodit.

Tästä syystä ensimmäiselle näkymälle tehtiin muutos, jossa piirakkakaavio korvattiin pylväsdiagramilla, jossa tietyn virheen lukumäärä näkyy selkeästi (kuva 23).

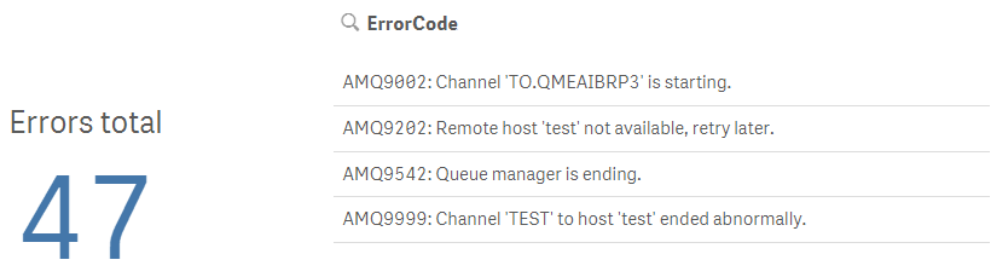


Kuva 23. Ensimmäisen näkymän piirakkakaavio on korvattu pylväsdiagrammilla, jotta virheiden lukumäärä näkyy paremmin.

Lisäksi toiselle näkymälle lisättiin virheiden kokonaislukumäärä, joka muuttuu asetettujen rajoitteiden mukaan. Lisäksi toiselle näkymälle lisättiin lista, jossa näkyvät kaikki virhekoodit, jotka löytyvät asetettujen rajoitteiden sisältä (kuva 24).

Itse prosessin ohjeista pilotoinnin tuloksena tuli hieman vähemmän palautetta. Palautteen vähyyttä selittää varmasti se, että ohjeet ovat tässä vaiheessa kehitystä asetetut kyseisen tilanteen mukaisesti ja ne tulevat muuttumaan huomattavasti käyttöönoton edetessä ja ennakkotilanteiden lisääntyessä.

Jatkokehityksen osalta, ennakkotilanteiden lisääntyessä, näkymälle tulisi lisätä liikennevalotyyppinen ratkaisu, josta käyttäjällä olisi mahdollista nähdä suoraan, vaatiiko tilanne mahdollisia jatkotoimia, ja näin ollen helpottaa entisestään näkymän käyttöä.



Kuva 24. Toiselle näkymälle palautteiden ansiosta kehitetty virheiden lukumäärää ja virhekoodeja kuvaavat kentät.

7.7 Prosessin jalkautuksen suunnittelu

Suunniteltuamme ja kehitettyämme prosessin sekä saadessamme prosessiin kuuluvat työkalut, kuten näkymän, jonomanagerit, datakeräimen ja datan suodatuksen yhteistoiminnan sille tasolle, että prosessi voitaisiin ottaa käyttöön, tuli seuraavaksi suunnitella prosessin jalkautus. Prosessi tultaisiin ottamaan käyttöön jatkuvien palveluiden teknisen hallinnan tiimissä ja asiakkuuden keskitetyssä valvontatiimissä. Tiimit kokonaisuudessaan ovat samaa kokoluokkaa, mutta toisessa tiimissä jäsenten vaihtuvuus oli nopeampaa. Kustannustehokkain ratkaisu oli siis järjestää yksi jalkautuskoulutus, johon osallistuvat molempien tiimien kaikki paikalle kykenevät jäsenet. Varmistuaksemme jalkautuksen onnistumisesta yhdessä yksittäisessä tilaisuudessa järjestimme prosessin pilotoinnin pienemmälle ryhmälle, jossa keräsimme viimeisiä mielipiteitä lopullista jalkautusta varten. Tärkeää oli saada tiimikohtaisia mielipiteitä prosessin kyvykkyydestä palvella tiimin päivittäistä toimintaa. Epämieluisa tai liikaa yksityiskohtia sisältävä prosessi kärsii ajan kuluessa satunnaisesti valikoitujen kohtien poishioutumisesta, mikäli esimerkiksi prosessi vaikuttaisi työläältä eikä jalkautuksessa otettaisi huomioon prosessin vaatimaa työmäärää tai selitettäisi tarkasti jokaisen välivaiheen tarkoitusta prosessille.

Koska minkään prosessin toiminnasta ei voida varmistua vain pelkän jalkautuksen perusteella, oli tärkeää järjestää tilannepalavereita, mitkä suunnittelimme pidettäviksi päivittäisen aamupalaverin yhteydessä. Vähintään kerran viikossa käytäisiin läpi suoritetun

valvonnan kohtaamat uudet virhetilanteet kokonaisuutena ja uudesta kohdatusta ratkaisemattomasta tilanteesta pystyisi pyytämään resursoitua ratkaisuapua muulta tiimiltä. Näin prosessissa kohdatut virhetilanteet ja prosessiin tehdyt lisäykset saadaan nopeasti koko tiimin tietoisuuteen ja varmistetaan prosessin jatkuva toiminta sekä kehitys.

8 Johtopäätökset

Insinöörityön tavoitteena oli suunnitella ja jalkauttaa jonomanagereiden kirjoittamien lokien lokiseurantaa monitoroiva prosessi kahden jatkuvien palveluiden teknisen hallinnan tiimin käyttöön. Suunniteltavan prosessin tavoitteena oli oikein käytettynä helpottaa kummankin tiimin työtaakkaa ja visualisoida monitorille sanomavirtojen tilannekuva niin lähihistoriaan kuin reaaliajassa. Monitorille analysoitavien sanomavirtojen mahdolliset virhetilanteet ja niiden aiheuttamat vikatilanteet auttaisivat tiimiä valitsemaan oikea-aikaisesti oikean toimenpiteen, jotta ongelmanratkaisu olisi proaktiivista pelkän tarkkailun sijaan.

Prosessin suunnittelun pääkriteerit käytiin läpi yhdessä suunnitteluun osallistuneen analytiikkatiimin kanssa. Kävimme lävitse prosessissa käytettävien työkalujen pääpiirteitä, kuten QlikView-näytön ominaisuuksia ja ympäristöt, johon kyseiset näytöt implementoitaisiin. Järjestimme erilliset tapaamiset niin projektin kehityksen käynnistämiselle kuin jatkokehitykselle. Koimme saaneemme enemmän kuin riittävän tuen prosessin kehityksen aloitusvaiheessa yritykseltä.

Aloitimme prosessin suunnittelun analysoimalla projektin kriittiset pisteet ja käymällä lävitse prosessissa käytettävät työkalut. Osa työkaluista oli oman työhistoriamme kautta tuttuja, joten niiden yhteistoiminnan tutkiminen tuntui mielekkäältä. Tutkimme myös, miten muissa teknologiayhtiössä jatkuvien palveluiden toiminta oli toteutettu. Prosessissa käytettävät työkalut määräytyivät jo ennen prosessin suunnittelun aloittamista jo olemassa olevien toteutusten myötä, joten myös valvontaprosessin suunnittelu tuli toteuttaa annettuja työkaluja hyödyntäen. Valinnaisuutta toteutukseen toivat itse näkymän suunnittelu ja datakeräyksen periaatteet sekä prosessin hyödyntäminen jatkuvien palveluiden tiimeissä.

Prosessia suunnitellessamme perehdyimme jatkuvien palveluiden teknisen hallinnan tiimien päivittäiseen toimintaan ja työnkuvaan. Saimme insinöörityön aiheen tilaustyönä

yritykseltä, joten tarve prosessille oli olemassa. Selvittäessämme tiimien työnkuvaa huomasimme, että molemmilla tiimeillä on käytössään useita eri työkaluja päivittäisten toimintojen ja tarkistusten tekemiseen. Tämä korostui etenkin yleisellä teknisen hallinnan tiimillä, sillä tiimin valvontavastuu jakautui lukuisille eri asiakkaiden ympäristöille. Prosessista haluttiin selkeä ja nopeasti sisäistettävä sekä riittävän yksinkertainen. Otimme myös huomioon, että molemmissa tiimeissä asiakasympäristön valvonnan päävastuu oli kiertävä, joten laaja ja kattava dokumentointi oli tarpeen. Tämä johti lopulta myös suunnittelumme ohjautumiseen kohti palvelutietämyksen hallintajärjestelmän käyttämistä osana prosessia, sillä molemmilla tiimeillä oli sen käyttöön vaadittava infrastruktuuri valmiina ja molemmat olivat kehittäneet omia hyväksi havaittuja toimenpiteitä sen ylläpitämiseen.

Prosessin toiminnallisuuden näkökulmasta näkymän kehitys oli tiimille tärkein. Liiketoiminnallisen hyödyn maksimoimiseksi näkymästä oli tärkeää saada mahdollisimman informatiivinen. Näkymän tuli olla selkeä, jotta prosessin toimintojen suorittaminen ei veisi aikaa muilta toiminnoilta. Näkymästä tehtiin kaksitasoinen: ensimmäisessä näkymässä käyttäjälle annettaisiin yleisnäkymä ympäristön tilanteesta ja toisessa näkymässä käyttäjällä olisi mahdollista selvittää virheiden juurisyy tarkemmin etenkin ajankohdan ja virheiden laadun perusteella.

Prosessin kehityksessä otimme huomioon ennakkotilanteiden puutteellisuuden. Lähtötilanteessa emme voineet kartoittaa kaikkia mahdollisia virhetilanteita, sillä kaikkia mahdollisia asiakastoteutuksia ja niiden kuvauksia ei ollut vielä toteutettu. Tämän insinöörityön aikana määritetyt prosessin ohjaamat toimenpiteet tulisivat kehittymään virhetilanteiden lisääntyessä.

Prosessin ja kehitetyn näkymän pilotointi ja arviointi toteutettiin kohdetiimien jäsenistä kasatun testiryhmän avulla. Pilotoinnin ja arvioinnin tavoitteena oli saada mahdollisimman paljon palautetta itse näkymästä ja sen toiminnasta prosessin tukena. Pilotoinnin tuloksena saatiin hyviä kehitysehdotuksia näkymän toiminnallisuuteen liittyen. Kehitysehdotuksia puntaroitiin ja niistä osa lisättiin lopulliseen jalkautettavaan versioon.

Insinöörityön lopputuloksena oli prosessi ja prosessin tueksi rakennettu QlikView-näkymä. Tavoitteisiin päästiin kehityksen osalta, mutta prosessin jalkauttaminen ei aikataulun takia onnistunut vaan se siirretään toteutettavaksi tulevaisuudessa.

Jalkautuksen myötä prosessi olisi ollut osana teknisen hallinnan tiimien päivittäisiä valvontarutiineja. Prosessi on suunniteltu häiriönhallintaa ja virhetilanteiden valvontaa varten, joten sen lopullinen päätarkoitus on toimia liiketoimintaa häiritsevien laajojen häiriöiden ehkäisemisessä ja integraatiototeutuksen häiriöstä toipumisen nopeuttamisessa.

Prosessin käyttöönotto olisi tuonut helpotuksia resursointiin myös valvonnan osalta – ainakin visuaalisesti. Virhetilanteiden eriytetyn näkymän toteuttaminen on myös asiakasraporttien luomisen kannalta edullinen, sillä suoraa tilannedataa on helpommin käsillä. Tilanteen selventäminen helpottuisi myös henkilölle, joka ei muuten ole tekemisissä kyseisen toteutuksen kanssa.

Prosessin kehitys tulevaisuudessa on pitkälti kiinni siitä, kuinka paljon virhetilanteita esiintyy ja kuinka niistä raportoidaan palvelutietämyksen hallintajärjestelmään. Ihannetilanteessa prosessin avulla pystytään ennakoimaan tulevia virhetilanteita ja prosessille määritellyillä toimenpiteillä virhetilanteet saadaan ennaltaehkäistyä.

Häiriönhallintaa ja virheenselvitystä reaaliaikaisesti tekevä prosessi ei kehittyessään kuitenkaan voisi syrjäyttää täysin inhimillistä valvontakoneistoa. Prosessi on suunniteltu asiakasta varten, ja niin kauan, kun asiakkaana on ihminen, tarvitaan myös ihminen prosessoimaan asiakkaan tarpeet.

Lähteet

1 David S. Linthicum. 2003. Next Generation Application Integration: From Simple Information to Web Services. Addison Wesley.

2 Jalote, Pankaj. 2005. An Integrated Approach to Software Engineering. Springer US.

3 Understanding Enterprise Application Integration - The Benefits of ESB for EAI. https://www.mulesoft.com/resources/esb/enterprise-application-integration-eai-and-esb#point-to-point. (Luettu 25.10.2016.)

4 Norton, Sarah., Waldman, Nell. 2011. Canadian Content. Nelson College Indigenous.

5 Office of Government Commerce. 2007. Introduction to the ITIL service lifecycle. The Stationery Office.

6 Frontiers in Massive Data Analysis. 2013. Committee on the Analysis of Massive Data; Committee on Applied and Theoretical Statistics; Board on Mathematical Sciences and Their Applications; Division on Engineering and Physical Sciences; National Research Council. ISBN:978-0-309-28778-4.

7 Gorlatch, Sergei. Danelutto, Marco. 2007. Integrated Research in GRID Computing. Springer US.

8 IBM MQ. http://www-03.ibm.com/software/products/en/ibm-mq. (Luettu 12.7.2016.)

9 Logstash Introduction. https://www.elastic.co/guide/en/logstash/current/introduction.html. (Luettu 12.7.2016.)

10 Parsing Logs Using Logstash. https://qbox.io/blog/parsing-logs-using-logstash. (Luettu 22.10.2016.)

11 QlikView Guided Analytics. http://www.qlik.com/us/products/qlikview. (Luettu 12.7.2016.)

12 Helpdesk Management Dashboard. https://s-media-cache-ak0.pinimg.com/originals/4c/db/11/4cdb11fc33c65df79ba11672591f38f9.jpg. (Luettu 22.10.2016.)

13 Error logs on Windows, UNIX and Linux systems. http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.mq.tro.doc/q039560_.htm. (Luettu 22.10.2016.)

14 Incident Management. https://computingnebula.files.word-press.com/2010/07/im.jpg. (Luettu 4.11.2016.)

15 Implementing a queue manager alias and cluster with WebSphere MQ. http://www.ibm.com/developerworks/websphere/library/techarti-cles/1212_bhat/1212_bhat.html.

16 Pover, Karl. 2013. Learning Qlikview Data Visualization. Packt Publishing Ltd.

17 Logstash. https://wikitech.wikimedia.org/wiki/Logstash. (Luettu 13.7.2016.)

18 Jatkuvat palveluthttp://www.hiqfinland.fi/Integraatiopalvelut/Jatkuvat-palvelut/. (Luettu 3.10.2016.)

19 Liiketoimintaprosessien kehittäminen. http://www2.amk.fi/digma.fi/www.amk.fi/opintojak-sot/0303012/1106227851022/1106577077518/1107020129145/1149533442477.html. (Luettu 3.10.2016.)

20 Anonyymi. Chief Executive, suppl. Guide to Enterprise Business Solutions (1999): 24-25.